FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Towards Interpretable Unbiased Behavioral Pattern Recognition

Leonardo Machado

Mestrado em Engenharia de Software

Supervisor: Doutor Hugo Sereno

July 26, 2018

Towards Interpretable Unbiased Behavioral Pattern Recognition

Leonardo Machado

Mestrado em Engenharia de Software

July 26, 2018

Abstract

Behavioral pattern recognition has long been one of the essences of human behavior. We collect information over time and turn evident patterns into knowledge. With the evolution of information technology, we have been able to collect much more information than we can naturally comprehend. But given our deep rooted desire to understand and recognized new patterns in everything, we have created a scientific field that focuses on creating software algorithms that are capable of making predictions based on what they have learned from this new information - this field is called Machine Learning.

In the past few decades, researchers have managed to make astonishing progress in creating faster and more powerful predictive algorithms, thanks to a specific type of Machine Learning algorithms called Artificial Neural Networks that can deeply process millions upon millions of data in a heartbeat. But as we seem to be getting better and better at making predictions, we still understand very little of what this information that we are processing means. These algorithms that we have created sure are good at learning from this information, but they have no means of explaining to us what they have learned.

The software industry has long been trying to use "learning" software to teach or tell us something learned from this information. In the health and wellness world of mobile applications for example, there are a special type of applications called "tracking applications". In these application, people write their life goals and log in whatever they are doing throughout the day. Then, as if they were "software coaches", these applications tell us what we could be doing to accelerate our progress towards our own goals. This seems to be a great example of software sharing patterns that they have learned from information that we gave them.

But unfortunately, this is a bit of a facade. What is really happening, is that these applications are simply labeling us according to some of our behavior, and then coach us by providing advices that work with most people that have also been identically labeled. Most of these labels are scientifically backed, but no two people are alike and these types of advices can lead to harm if they are not sufficiently accurate. It seems here that, we are in fact not actually transmitting all of the knowledge learned from these predictive algorithms - we are mostly shaping and generalizing some of the information.

As a mean for approaching these problems, the purpose of this thesis is to create two rudimentary methods for the extraction of behavioral patterns learned by predictive artificial neural networks. The two methods combined, will allow for the understanding of how specific behaviors are influencing the prediction of a target behavior by providing two metrics: a "weight rank", which indicates the influence of each behavior on the prediction of the target behavior; and an "average linear correlation direction" which, in case of there being a clear linear correlation between a behavior and a target behavior, indicates if the average relationship between these two behaviors is proportional or inverse.

To test the created methods, we extracted the behavioral patterns detected by a long short-term memory network responsible for predicting the levels of PM2.5 concentration in Beijing's

atmosphere during the peak of the winter of 2010.

By the end of this thesis, we will have conceptually proven that we can in fact extract two behavioral patterns from an artificial neural network with a precision of at least 60%. Still, there is enough room for performance improvements which can surely result in more in-depth pattern extraction methods.

This will hopefully help pave a new path towards the discovery of new and more complex concepts and patterns, and a greater analysis of previously thought-to-be overwhelming amounts of data.

Keywords: pattern recognition, behavioral patterns, machine learning, artificial neural networks

Resumo

O reconhecimento de padrões comportamentais sempre foi uma das essências do comportamento humano. Informação é recolhida ao longo de um dado periodo de tempo e padrões evidentes são depois transformados em conhecimento. Com a evolução das tecnologias de informação, tem havido cada vez mais fontes de informação que tornam difícil a total compreensão de toda esta informação criada.

De forma a colmatar esta falha, foi criado um campo científico que se foca na elaboração de algoritmos de software capazes de realizar previsões de acordo com a informação apreendida - este campo científico denomina-se Machine Learning.

Durante as últimas duas décadas, investigadores têm vindo a fazer grandes progressos no desenvolvimento de algoritmos preditivos, particularmente com a criação de um tipo particular de algoritmo de Machine Learning chamado de "Redes Neuronais Artificiais", capazes de processar grandes quantidades de dados muito rapidamente. Mas há medida que nos vamos tornando cada vez melhores a fazer previsões, continuamos a não ser capazes de compreender todo o conhecimento subjacente a esta informação que processamos. Isto deve-se ao facto que estes algoritmos que criamos são bons a aprender padrões através de informação, mas infelizmente não possuem bons métodos para transmitir e explicar estes mesmos padrões.

Já há algum tempo que a indústria de software tem tentado usar software "que aprende" para nos ensinar novos pontos de vista relativamente a dados existentes. Por exemplo, na àrea de aplicações de saúde e do bem-estar, existem um tipo particular de aplicações chamadas "tracking applications" em que os utilizadores introduzem os seus objectivos quotidianos e depois, fazendo uma espécie de um diário, indicam as actividades que foram fazendo ao longo do tempo. De seguida, as aplicações agem como se fossem "treinadores", dizendo ás pessoas o que é que elas poderiam estar a fazer para conseguir chegar aos seus objectivos de uma forma mais eficiente. Isto aparenta ser um óptimo exemplo de uma situação em que um programa partilha padrões compreendidos, através de dados previamente providenciados.

Infelizmente, isto é perdominantemente uma fachada. Na realidade, o que se está a passar, é que estas aplicações apenas se limitam a agrupar pessoas de acordo com alguns comportamentos que elas apresentam. E depois guiam-nas dando conselhos que são apropriados para a maioria das pessoas que apresentam comportamentos semelhantes. Os métodos utilizados para agrupar as pessoas são geralmente comprovados cientificamente, mas não existem duas pessoas iguais e estes tipos de conselhos podem causar dano caso não sejam suficientemente precisos. Assim sendo, na verdade, nós não estamos a transmitir todo o conhecimento retirado destes algoritmos preditivos - estamos maioritariamente a moldar e a generalizar parte da informação.

Com o objectivo de resolver estes problemas, esta tese pretende abordar a criação de dois métodos rudimentares para a extracção de padrões comportamentais apreendidos através de redes neuronais artificiais preditivas. A combinação dos dois métodos permitir-nos-á perceber como é que um determinado comportamento está a influenciar a previsão de um comportamento alvo,

utilizando para isso duas métricas: um "ranking de peso" que indica a influência que cada comportamento tem na previsão de um comportamento alvo; e uma "direcção média de correlação linear" que, no caso de haver uma clara correlação linear entre um comportamento e um comportamento alvo, indica se grande parte do relacionamento entre os dois é proporcional ou inversa.

De modo a testar os métodos criados, extraimos os padrões comportamentais detectados utilizando uma rede neuronal Long Short-Term Memory para prever a concentração de particulas PM2.5 na atmosfera de Pequim durante o pico do Inverno de 2010.

No final desta tese, provamos que pudemos de facto extrair dois padrões comportamentais de uma rede neuronal artificial com uma precisão de pelo menos 60%. Ainda assim, existe espaço para futuras melhorias que poderão resultar na criação de métodos de extracção de padrões comportamentais mais complexos.

Esperamos que isto abra novos caminhos em direcção à descoberta de novos e mais complexos padrões e conceitos, e também uma melhor análise de dados.

Keywords: reconhecimento de padrões, padrões comportamentais, machine learning, redes neuronais artificiais

" Dedicated to those who stood by me. "

Contents

List of Figures

List of Tables

Abbreviations

TSA	Time Series Analysis
TSF	Time Series Forecasting
ANN	Artificial Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory Network
RMSE	Root-Mean-Square Error

Chapter 1

Introduction

Contents

Pattern recognition has been considered by many, the essence of human behavior - we act with learned behavior and we react based on what we know. [26] With some of the most recent information technology advancements, we have been able to gather a *large amount of new external data* from sensors, software transactions, business transactions, records and many other sources. But with it, came one of the great challenges that we have been facing in this information era: *how can we process this data and turn it into patterns and knowledge?*

Given our deep rooted desire to understand and recognized new patterns in everything, we created the scientific field of Machine Learning to help us with this problem. By allowing computers to use statistical methods, Machine Learning's software algorithms can process, detect patterns and learn from virtually any type of data. In the predictive branch of this field, exist a series of algorithms which - by processing and autonomously recognizing patterns on a specific dataset - can predict future values with a certain degree of confidence. These predictions can then be used as knowledge when compared with other information or just as standalone insight.

But how often do we actually use it to help us understand new behavioral patterns?

1.1 Computer Science in Behavioral Pattern Recognition

We comprehend ***behavior*** as being: *the way in which something functions or operates*. Therefore, when any entity - whether it is a person, an object or a grouping - presents an action at any given point in time, we refer to it as that entity's behavior. Since time is one of the defining characteristics of any behavior's identity, we then perceive ***behavioral patterns*** as being *a replicable set of behaviors which exist in a specific time-frame*.

Machine Learning, has long been focused on recognizing behavioral patterns and making predictions based on them: from making stock-value predictions to weather forecasting, to even predicting the amount of pollution in the atmosphere based on multiple natural factors. These particular types of predictive problems are called ***Time Series Forecasting*** problems. Time Series Forecasting focuses on making predictions based on data which has a chronological order - called *time-series data*.

Time Series Forecasting (TSF) can be tackled in many different ways - using a plethora of different algorithms; each with its own purpose, parameters and performance [19]. Most recently, in *Deep Learning* (a subfield of Machine Learning), *Artificial Neural Networks* have been used to dynamically adjust to constant flows of new data and make new forecasts based on new and old data combined.

1.2 Large Amounts of Forecasting, Not Enough Analysis

But as we become better and better at making faster and more accurate predictions, computer science research seems to have been **focusing fairly heavily on predictive performance in comparison with the focus given to analyzing patterns in time-series data**. Doing so is obviously beneficial to help with most types of prevention or opportunity ceasing that may be found in most scientific fields, e.g. finance, meteorology, engineering; but many others would also benefit dearly from being able to **understand how these predictions are being made and which behavioral patterns are related to them**. For example, if we look at any field that relies on the scientific method - e.g. medicine, biology, ecology, astronomy:

> *the ability to take large amounts of seemingly unrelated data, process it, make good predictions with it, and then extract new previously undetected and unknown behavioral patterns, would allow for new background research and hypothesis construction on previously thought to be scattered and overly-extensive data.*

This could lead to amazing new scientific discoveries and another step towards better human-machine interactions and shared knowledge.

But first, in order to do so, we should focus our attentions on ***Time Series Analysis (TSA)***. As the name implies, Time Series Analysis refers to when we look at a time-series dataset and try to *comprehend how values are changing, at what rate are they changing, and how they can be replicated*.

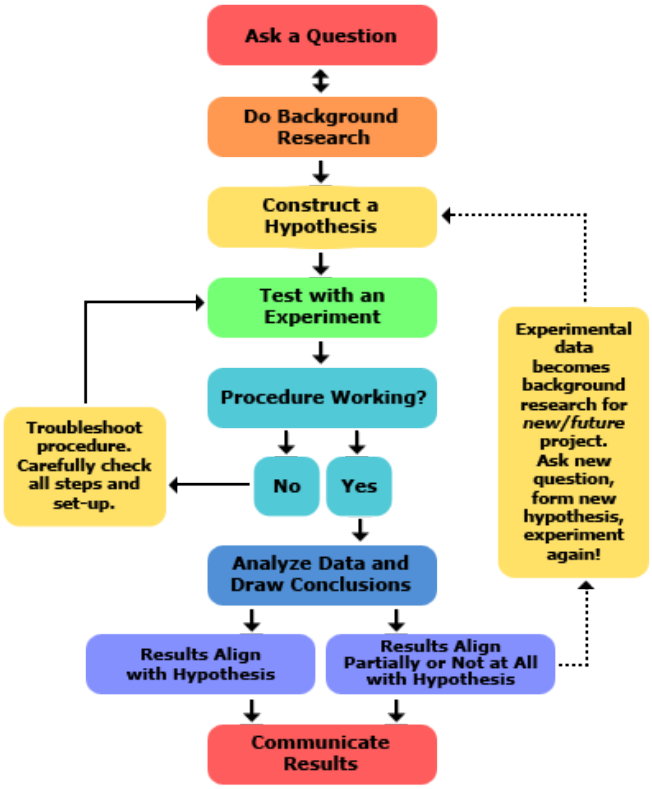

Figure 1.1: Steps of the Scientific Method [6].

Throughout history, the majority of time-series analysis has either been done through visual analysis of graphs or through the implementation of mathematical formulas to assess any type of behavioral pattern. Given the computational and scientific advancements made in the past couple of decades, these practices may seem a bit unorthodox. Computers have been used to achieve fascinating results in data processing, pattern recognition and forecasting, so:

Question 1: ***Could we use computer science to help us take time series analysis and behavioral pattern recognition a step further?***

1.3 The One-Size-Fits-All Paradigm

Before searching immediately for a solution to this problem, we decided to look for ways in which the software industry have tried to use "learning" software to provide us with knowledge learned from this new information. So we asked ourselves:

Question 2.1: ***How is the software application industry using behavioral data and Time Series Analysis to help us learn?***

Question 2.2: ***Can the approach used be improved upon in any way?***

To answer these questions, we searched for mobile and desktop applications which were concerned with recording any type of behavior, whether that was human or non-human, and then

sharing any type of knowledge learned from those behavior. We found a category of applications called ***tracking applications***. These applications are often inserted in the health and fitness domains. They allow people to track exercise plans, track meals, and learn how their activities affect their dietary and life goals. As an attempt to aid users on the advancement towards their goals, some of these applications provide behavioral advices and tips e.g. MyFitnessPal - a calorie counter application.

However, these applications are not actually using individual behavioral analysis to learn what is best for each user, most of these applications are instead just **labeling users and advising them according to their labels**. For example, by inquiring users about their: height, age, weight and relative activity level; some of these applications categorize their users, place them in groups with people that have similar measurements and advise them according to each group's common behavior and characteristics [15, 17]. This leads to a paradigm called the **one-size-fits-all paradigm**. When adopting this *inherently biased paradigm*, each individual is advised to follow a specific path that may be suited for the majority of *identically labeled individuals* but is not well-suited for their specific needs. That can lead to three significant problems:

1. Over-extended time frame for expected results;
2. Unreachable results;
3. Unexpected neglect or stress on overlooked details.

All of these problems cast clear negative prospectives for companies that use this paradigm, but for those implementing it in the field of health and well-being there is a chance that it **can lead to actual physical harm** for the user.

1.4 Primary Goals

This thesis was developed with the primary objective of contributing to the body of knowledge in software engineering as follows:

1. **Creating the foundation for a software-based method which extracts behavioral patterns learned by advanced predictive algorithms and presents them in an easy to comprehend format**

As we will address in chapter 4, this method will consist in the extraction of two patterns that will enable a basic comprehension of the correlation between behaviors.

1.5 Scientific Contributions & Motivation

Up until this point, we have established that there exist two significantly unexplored perspectives in the behavioral pattern recognition field of *software engineering*.

As was suggested on section 1.2, the **usage of outdated TSA methodologies has led to a very slow development of the field** in comparison to how much research and development has been done in the past decades in the closely related area of *forecasting algorithms*. But the biggest contribution to be made is not as directly related to TSA as it is with the more wide-ranging fields of science and the scientific method; citing part of section 1.2:

> *"(...) the ability to take large amounts of seemingly unrelated data, process it, make good predictions with it, and then get new previously undetected behavioral patterns, would allow for new background research and hypothesis construction on previously thought to be scattered and overly-extensive data."*

The formulation of a method which could allow for a more *synergistic collaboration* between **traditional scientific method execution and computer data analysis** could help bridge another gap between conceptual and thought-to-be cluttered data, and empirical knowledge. By setting forth momentum towards finding an answer to Q.1. we could become able to extract new knowledge from accurate forecasts and then test that knowledge in the hopes of reaching new observable facts.

On a different side of the spectrum we have Q.2. which, as aforementioned, is much more specific in nature. The contributions made from solving this question in particular could not only help **prevent physical harm made from neglected behavioral details**, but it could also be used as a mean of **looking at behavior in an unbiased manner so that new raw insight could be collected** from behavioral correlations, and individual human needs could be best met.

In summary, approaching the whole scope of the problem could contribute to the fields which are encompassed in this thesis: software engineering and data science; but the technology developed off of this thesis' contributions can create potential advancements in many other scientific fields.

1.6 Document Structure

This document will have the following structure:

1. **Chapter 1 - Introduction:** a brief introduction will take place, in which the reader will acquire a general idea of the problems that will be addressed and the goals and contributions that motivate this thesis.

2. **Chapter 2 - Background:** a series of concepts will be introduced, which will allow for the reader to fully understand the solution developed and the discussions that will exist throughout.

3. **Chapter 3 - State-of-the-Art:** existing competing solutions will be discussed so that the initially raised questions and problems can be validated and a solution proposal can be formulated.

4. **Chapter 4: - Problem Statement:** the initial problems will be fully delved into so that their scopes are well understood. A solution proposal will be formulated. The main goals will be set and an evaluation strategy will be created so that the reader can understand how the authors defined a successful implementation scenario. Furthermore, some possible obstacles that might be encountered during the development process of the work will be addressed and the main contributions of the work will be revisited.

5. **Chapter 5 - Implementation Foundation:** the start of the implementation phase which begins with a discussion of which algorithms, data and technologies will be used for the development of the solution.

6. **Chapter 6 - Forecasting Algorithm:** the forecasting algorithm from which the behavioral patterns will be extracted is created, and all of its implementation details are discussed.

7. **Chapter 7 - Behavioral Pattern Extraction Methods:** the methods which will extract the behavioral patterns discussed on chapter 4 are implemented and tested.

8. **Chapter 8 - Conclusions:** the results are discussed accompanied by a discussion of how they can be improved upon in future work, and general comment is made on the work developed.

As a quick disclaimer, this document was structured in a way that allows the reader to grasp new content and understand how the problems and the solutions were formulated, in a progressive and incremental manner. For this purpose, it was better to sometimes group some individual research segments according to subject and not chronologically. This will intermittently result in small jumps between information presented in different sections or chapters. Still, every time that this happens, the reader will be notified and quickly acquainted with every relevant chronological event leading to that point in the thesis - further details will then be disclosed when the reader reaches the referenced section or chapter.

Chapter 2

Background

Contents

Further along this thesis, particularly from the start of the implementation process (chapter 5 onwards), the reader is expected to be fairly acquainted with specific subject matters, particularly: time series, machine learning algorithms and artificial neural networks. As we go through this chapter, the familiarity with these topics will allow for a deeper understanding of the problems and solutions discussed, as well as a more fluent and coherent reading.

In the end, the reader is expected to be able to comprehend the aspects which need to be taken into consideration when searching for a possible solution for this thesis' problems, respectively: how to approach time series problems, what to consider when choosing machine algorithms and what is the purpose of artificial neural networks.

We will be starting with a revision of Behavioral Pattern Recognition and Time Series as they were first addressed in chapter 1. Briefly, we will also mention basic technical terms used when talking about data and data manipulation.

2.1 Behavioral Pattern Recognition

Behavioral Pattern Recognition is a very broad term which can be applied to many distinct fields and in its breadth we can recognize a compound definition of multiple different terms. But for the work developed, we just need to settle on a consensus on what it is and how do we perceive it.

Let us take a look at the first term used in defining the field: **Behavior**.

Behavior can be defined as the **manner in which something functions or operates**, regardless of the subject's features. When any entity presents an action at any given point in time, we refer to it as that entity's behavior.

The following two terms can be conjugated as a compound term: **Pattern Recognition**.

Pattern recognition is the act of **recognizing patterns or regularities in a given scenario**. In Computer Science, the field of Pattern Recognition is concerned with providing a reasonable answer for all possible data and to classify input data into objects or classes based on certain features [38].

Finally, by compiling both of these terms together we can define **Behavioral Pattern Recognition** as being the action of **recognizing patterns and regularities in how an entity functions and operates**. From this point onwards, this will be the definition used when referring to Behavioral Pattern Recognition.

2.2 Technical Language for Data

When talking about data and data manipulation there are a series of technical terms that differ from field to field. In this thesis, we will be using the terminology most commonly found in computer science.

2.2.1 Attributes & Instances

In computer science, when we talk about datasets rows and columns, they often carry a different meaning.

- **Rows** are referred to as **instances** or as **observations**;
- **Columns** are referred to as **attributes** or **features** of an instance or observation;

2.2.2 Models & Algorithms

Despite often being used interchangeably, models and algorithms are different things, even though one is the product of the other. In computer science, particularly when talking about data science and machine learning algorithms: a **model** is the product of the processing that an **algorithm** does on **data**.

$$Model = Algorithm(Data)$$

The formula above is a good way of visualizing this relationship. Model ultimately is a representation that the algorithm created from the data that it processed.

2.3 Time Series Overview

Time series or *time series model* is a **series of data points indexed, listed or graphed in time order**. Even though there are time series with sequences taken in disperse points in time, time series sequences are usually taken at successive equally spaced points in time. Despite most time series datasets having a time signature which identifies the point in time in which each instance was recorded, it is also common for them to have none. As long as the time gap between instances was previously defined, *time series datasets can have implicit time signatures* - e.g. if a dataset is comprised of a single stock value attribute, if told to, we can imply that each new instance was recorded one day after the previous one, making it an implicitly daily time series.

Figure 2.1: Time Series of IBM's Common Stock Value on June 1st 2018 [13].

Time series are very frequently plotted via line charts as seen on figure 2.1. They can be used virtually on any domain of applied science and engineering which involves temporal measurements [19]. It is important to know that there are two distinct types of models:

1. **Univariate:** refers to a time series that consists of **single (scalar) observations** recorded sequentially over equal time increments [1];
2. **Multivariate:** refers to a time series that consists of **multiple observations** recorded sequentially over equal time increments [30].

They differ in a simple aspect, but as we go further along the thesis, this difference will weigh heavily on the choices made regarding the algorithms used in the final solution - because particular algorithms can only work with one type of model and others work better with another.

Perhaps the most useful way of defining a time series is by decomposing it into 4 distinct constituent parts:

1. **Level:** The baseline value for the series if it were a straight line;
2. **Trend:** The optional and often linear increasing or decreasing behavior of the series over time;

3. **Seasonality:** The optional repeating patterns or cycles of behavior over time;

4. **Noise:** The optional variability in the observations that cannot be explained by the model.

All time series have a **level**, *most* have **noise**, and *some* have **trends and seasonality**. These constituent components can be thought to combine in some way to provide the observed time series - e.g. they may be added together to form a model as follows:

$$y = level + trend + seasonality + noise$$

Assumptions can be made about these components both in behavior and in how they are combined, which allows them to be modeled using traditional statistical methods [5].

Now that we have grasped the necessary core aspects of time series, the following subsections will be presenting the two main methods for knowledge extraction of time series data.

2.3.1 Time Series Analysis

The first method is called **Time Series Analysis** (TSA). As was previously mentioned, TSA attempts to extract knowledge from time series by analysing possible patterns that may exist within the data [30]. TSA involves developing models that best capture or describe an observed time series in order to understand the underlying causes. This field of study seeks the "why" behind a time series dataset [5]. Some professionals also argue that the reason for which TSA was created was because:

> *"Time series analysis accounts for the fact that data points taken over time may have an internal structure (such as autocorrelation, trend or seasonal variation) that should be accounted for [30]."*

2.3.2 Time Series Forecasting

In an attempt to go beyond analysis, **Time Series Forecasting** was created. Forecasting involves **taking models fit on historical data, and using them to predict future observations**. An important distinctive characteristic of forecasting, is that the future is completely unavailable and must only be estimated from what has already happened.

> *"The purpose of Time Series Forecasting is generally twofold: to understand or model the stochastic mechanisms that gives rise to an observed series" (Time Series Analysis) "(...) and to predict or forecast the future values of a series based on the history of that series." (Time Series Forecasting)*

The *skill* of a time series forecasting model is determined by its *performance at predicting the future*. This is often at the expense of being able to explain why a specific prediction was made, confidence intervals and even better understanding the underlying causes behind the problem.

Therefore, in order to reduce these expenses, there was created a method which through dialogue and constant questioning, can help one decide how to create the best possible forecasting model.

1. **How much data do you have available and are you able to gather it all together?** More data is often more helpful, offering greater opportunity for exploratory data analysis, model testing and tuning, and model fidelity.

2. **What is the time horizon of predictions that is required?** Short, medium or long term? Shorter time horizons are often easier to predict with higher confidence.

3. **Can forecasts be updated frequently over time or must they be made once and remain static?** Updating forecasts as new information becomes available often results in more accurate predictions.

4. **At what temporal frequency are forecasts required?** Often forecasts can be made at a lower or higher frequencies, allowing you to harness down-sampling, and up-sampling of data, which in turn can offer benefits while modeling.

It is important to note that since Time Series Forecasting requires the comprehension of a time series model, it is **much easier to read a time series with equally spaced time instances**, and most algorithms created for forecasting require just that.

Finally we must also take into consideration that time series data **often requires cleaning, scaling, and even transformation**: data can be provided at a frequency that is too high to model or is unevenly spaced through time, requiring resampling for use in some models; there can be corrupt or extreme outlier values that need to be identified and handled; and there may even be gaps or missing data that need to be interpolated or inputed [5].

2.4 Machine Learning

Machine Learning (ML) is a combination of multiple scientific fields: artificial intelligence, statistical computing and data analysis. Its focus is the construction of algorithms which can learn iteratively and make predictions on data [34].

As new data is presented, these algorithms are able to adapt and build upon the knowledge that they had previously assessed, making them iteratively more efficient and reliable with time and training. Before diving into what types of machine learning algorithms there are, and what their purposes are, let us touch on a few common topics.

First of all, machine learning algorithms can perform any of these tasks:

- **Classification:** insert something into a category, *e.g. this is an orange*;

- **Regression:** determine the value of something, *e.g. tomorrow the stock value of Apple Inc. will be 194 USD*;

- **Clustering:** create inner-groups of something based on similar aspects, *e.g. here are all the small and green products from your shopping bag*;
- **Association:** correlate instantaneous events, *e.g. when someone buys flour, they also buy baking powder*.

These are all predictions, as in estimations of what is happening. When a machine learning algorithm is run, they create a **model**. A model, or a ***mapping function***, is a **representation of what the "true" underlying *facts* in the original data are**. These models are always an estimate of what might be happening, they are never a perfect representation of the actual mappings of the real solution because we can never fully assert anything - there is always some additional data that is not being accounted for.

2.5 Learning Styles

There is a multitude of different Machine Learning algorithms. Given the appropriate data, these algorithms create models of the problem by interacting with this data in a specific manner. We call this course of action: the algorithm's ***learning style***. There are three learning styles categories in which machine learning algorithms can fall into: supervised learning, unsupervised learning and semi-supervised learning.

2.5.1 Supervised Learning

The majority of practical machine learning uses supervised learning. Supervised learning is where you have input variables (X) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output.

$$Y = f(X)$$

The goal is to approximate the mapping function so well, that when you have new input data (X), your mapping function allows you to predict the output variables (Y) for that data. In this learning style we refer to input data as *training data*. It is called supervised learning because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process of a student.

1. The algorithm reads the first F percentage of data from the set;
2. Creates a prediction the remaining R percent;
3. Reads the actual values of the remaining R percentage of data;
4. Corrects its prediction criteria (as if it was a teacher telling it what they thought they did wrong);

5. Starts over from step 1. until performance level K is achieved.

Supervised learning problems can be further grouped into regression and classification problems.

- **Classification:** when the output variable is a category, such as Purchased or Sold or Spain or Italy;
- **Regression:** when the output variable is a real value, such as the amount of dollars or weight.

Common types of problems which can be seen as classification and regression problems include: recommendations and time series prediction respectively. Popular examples of supervised machine learning algorithms: *linear regression, random forest, support vector machines, artificial neural networks and naive bayes* [4].

2.5.2 Unsupervised Learning

Unsupervised learning is where you only have input data (X) and no corresponding output variables. The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data.

This is called unsupervised learning because unlike supervised learning, there is no correct answers and there is no teacher. Algorithms are left to their own devises to discover and present the interesting structure in the data.

Unsupervised learning problems can be further grouped into clustering and association problems:

- **Clustering:** you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior;
- **Association:** you want to discover rules that describe large portions of your data, such as people that buy A also tend to buy B.

Popular examples of unsupervised machine learning algorithms: *k-means and apriori* [4].

2.5.3 Semi-Supervised Learning

Semi-supervised learning problems exist when you have a large amount of input data (X) and only some of the data is labeled (Y). These problems sit in between both supervised and unsupervised learning. A good example is a photo archive, where only some of the images are labeled, (e.g. dog, cat, person) and the majority are unlabeled.

You can use unsupervised learning techniques to make best guess predictions for the unlabeled data, feed that data back into the supervised learning algorithm as *training data* and use the model to make predictions on new unseen data [4].

2.6 Parametric and Nonparametric Algorithms

We have seen that machine learning algorithms can learn in different ways: some learn by readjusting their models according to training data, and others simply search for a solution without any guidelines. But what and how these algorithms learn can vary immensely depending on which parameters they use to find underlying function forms on data.

2.6.1 Parametric Algorithms

Some algorithms are much faster than others. By making assumptions about the data, they can skip one of the most resource consuming steps of the learning process - creating custom function forms for better pattern recognition. But there is a caveat to this approach: assumptions can greatly simplify the learning process, but can also limit what can be learned. These algorithms, also known as **parametric machine learning algorithms**, simplify the function of the input to a known form, given by the data scientists who created it, and then simply stick to it. **No matter how much data you feed to a parametric model its *complexity* will remain the same**, i.e. it will always stick with the same approach and the same parameters.

An easy to understand functional form for the mapping function is a line, as is used in linear regression:

$$B0 + B1 * X1 + B2 * X2 = 0$$

Where B0, B1 and B2 are the *coefficients* of the line that control the intercept and slope, and X1 and X2 are two input variables. **Assuming the functional form of a line greatly simplifies the learning process**. Now, all we need to do is estimate the coefficients of the line equation and we have a predictive model for the problem.

Often the assumed functional form is a linear combination of the input variables and as such, parametric machine learning algorithms are often also called **linear machine learning algorithms**. The problem is, **the actual unknown underlying function may not be a linear function** - like a line for example. It could be very similar to a line and require some minor transformation of the input data to work right. Or it could be nothing like a line in which case the assumption is wrong and the approach will produce poor results.

In sum there are benefits and limitations to parametric machine learning algorithms. They can be a good choice given their:

- **Simplicity:** they are easier to understand and interpret;
- **Speed:** can learn from data very quickly;
- **Minimal Data Requirement:** they do not require as much data.

On the other hand they might not be the optimal solution due to their:

- **Constraints:** they are highly constrained to the specified form;

- **Limited Complexity:** they are more suited to simpler problems;
- **Poor Fit:** in practice, they are unlikely to match the underlying mapping function.

Popular examples of parametric machine learning algorithms: *logistic regression and perceptrons* [4].

2.6.2 Nonparametric Algorithms

On the other side of the spectrum, we have **nonparametric machine learning algorithms**. These are algorithms that do not make strong assumptions about the form of the mapping function. By not making assumptions, they are free to learn any functional form from the training data. Put simply, **as you feed more and more training data into a nonparametric model the algorithm will keep on "rethinking" the model and thus increasing its *complexity* over time**.

Nonparametric methods seek to best fit the training data in constructing the mapping function, whilst maintaining some ability to generalize to unseen data. As such, they are able to fit a large number of functional forms and are praised for their:

- **Flexibility:** they are capable of fitting a large number of functional forms;
- **Power:** they are not very constrained to assumptions about the underlying function;
- **Performance:** can create better performing models for prediction.

But their power and flexibility comes at a cost:

- **Require More Data:** require a lot more training data to estimate the mapping function;
- **Slower:** slower to train due to their large amount of parameters to train;
- **Overfitting:** can create models which are too broad and so, their predictions become harder to understand.

Popular examples of nonparametric machine learning algorithms: *decision trees, naive bayes, support vector machines and artificial neural networks* [4].

2.7 Bias and Variance

Supervised machine learning algorithms learn a model from training data. The goal of any supervised machine learning algorithm is to best estimate the mapping function (f) for the output variable (Y) given the input data (X). The mapping function is often called the ***target function*** because it is the function that a given supervised machine learning algorithm aims to approximate to. When comparing the predicted function with the target function, we obtain an error which can be broken down into three distinct parts:

- Bias Error
- Variance Error
- Irreducible Error

The ***irreducible error*** cannot be reduced, regardless of what algorithm is used. It is the error introduced from the chosen framing of the problem and may be caused by factors like unknown variables that influence the mapping of the input variables to the output variable. In this section we will be focusing on the two parts we can influence with our machine learning algorithms: The *bias error* and the *variance error*.

2.7.1 Bias Error

Bias are the simplifying **assumptions made by a model to make the target function easier to learn**. Generally parametric algorithms have a high bias making them fast to learn and easier to understand but generally less flexible. In turn they have lower predictive performance on complex problems that fail to meet the simplifying assumptions of the algorithms bias.

- **Low-Bias:** less assumptions about the form of the target function, *e.g. nonparametric algorithms*;
- **High-Bias:** more assumptions about the form of the target function, *e.g. parametric algorithms*.

Popular examples of **low-bias machine learning algorithms**: *decision trees, k-nearest neighbors and support vector machines*.

Popular examples of **high-bias machine learning algorithms**: *linear regression and Logistic Regression*.

2.7.2 Variance Error

Variance is the **amount that the predicted target function will change if different training data was used**. The target function is estimated from the training data by a machine learning algorithm, so *we should expect the algorithm to have some variance*. Ideally, it should not change too much from one training dataset to the next, meaning that the algorithm is good at picking out the hidden underlying mapping between the inputs and the output variables.

Machine learning algorithms that have a high variance are strongly influenced by the specifics of the training data. This means that the specifics of the training data influences the number and types of parameters used to characterize the mapping function.

- **Low Variance:** Suggests small changes to the estimate of the target function with changes to the training dataset, *e.g. parametric algorithms*;
- **High Variance:** Suggests large changes to the estimate of the target function with changes to the training dataset, *e.g. overly flexible nonparametric algorithms*.

2.7.3 Finding the Balance

The main goal of any algorithm optimization is to make the error as close to the irreducible error as possible; but as we can see, the Bias and the Variance errors are opposing forces on the search for this optimization. As we **decrease the bias** to make a more flexible function, we will be **increasing the variance** and thus creating more unpredictable models.

Though we will never know the actual underlying target function, we know that not all underlying functions work the same so we work to find the range in which the greatest decrease of variance creates the least increase of bias and vice-versa.

2.8 Fitting Models

The learning process that every machine learning algorithm is responsible for is called *induction*. When inducing, the algorithm is learning general concepts from specific examples given by the training data. The metric which allows us to understand if an algorithm is doing a good job at inducing a model, i.e. if the models created fit well on new unseen data, is called *Generalization*.

When determining whether a model is generalizing well or not, we weigh our decision on two factors:

*Is the model **overfitting** the data?*

OR

*Is the model **underfitting** the data?*

Ideally, when the generalization is perfect the induced function from the training data is exactly the same as the underlying target function. But this is an ideal scenario, the unknown nature of new data means that the majority of times generalization is not close to perfect, so we have to take into consideration how models are being overfitted or underfitted.

2.8.1 Overfitting

Overfitting happens when a **model is modeling the training data too well**. The model has learned the detail and random fluctuations in the training data to the extent that it negatively impacted the performance of the model on new data. This means that the random fluctuations in the training data are being picked up and learned as concepts by the model. The problem is that these concepts do not apply to new data and **negatively impact the models ability to generalize**.

Overfitting is more likely with *nonparametric and nonlinear models* that have more flexibility when learning a target function.

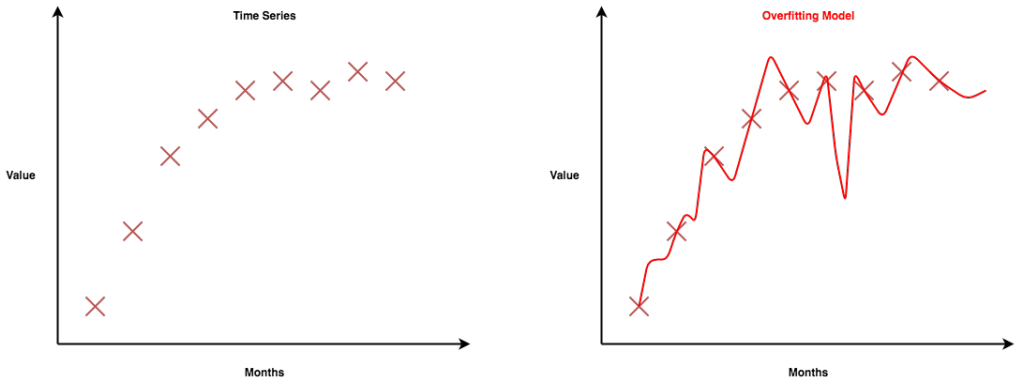

Figure 2.2: Example of an Overfitting Predictive Time Series Function (right graph).

2.8.2 Underfitting

Underfitting happens when a model **can neither model the training data nor generalize to new data**. An underfit machine learning model is not a suitable model and will be obvious as it will have poor performance on the training data. Underfitting is often not discussed as it is easy to detect given a good performance metric. The remedy is to move on and try alternate machine learning algorithms. Nevertheless, it does provide good contrast to the problem of concept of overfitting.

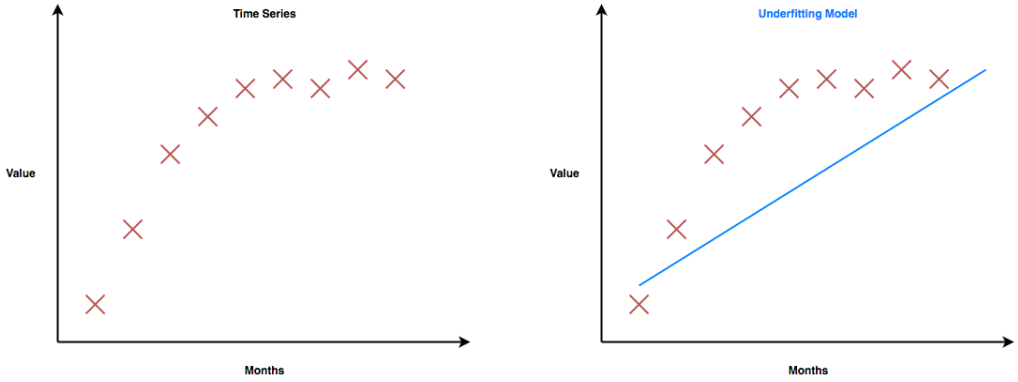

Figure 2.3: Example of an Underfitting Predictive Time Series Function (right graph).

2.8.3 Finding A Good Fit

A perfect model does not exist. When working on a predictive problem, we cannot get a model that perfectly fits the target function; this happens because of noise in the data, missing data, incomplete data and numerous other factors. What we can do is try to be as close to perfect as possible.

How do we work towards perfection?

First we need to create a ***baseline*** for our problem and then we work towards improving that baseline. A baseline will be an initial point of reference of how good the first prediction was. To determine the quality of a prediction in regression problems, which will be the majority of problems found in this thesis, we look for the **error between the estimated function from the training data and the target function extracted from the test data**. In most literature, the standard error for regression problems is the ***root-mean-square error (RMSE)***.

RMSE tells us how close are the predicted values from the actual target values. An RMSE of 0.0 means that every predicted value is the same as the actual target value - also known as a *perfect fit* [29]. The value of the RMSE for any prediction is an absolute value that is dependent on the predicted variable. It shows the standard deviation for how off the predictions are from the actual values that are being predicted. For example, if we are trying to predict the temperature of the weather tomorrow (in degrees Celsius) and our model has an RMSE of 5 it means that the predicted temperature values should be off by 5 degrees more or less.

Once we have a baseline, we can then change the:

1. **Data:** fill in missing data, reduce noise, remove unnecessary data;

2. **Parameters:** change the parameters to understand when does the algorithm work better;

3. **Hardware:** sometimes the hardware itself can be constraining the performance of the algorithm;

4. **Algorithm:** occasionally some algorithms are just not the best fit for that specific problems and need to be replaced;

Hopefully, after some reiterations we can create an algorithm that solves the problem, given its requirements.

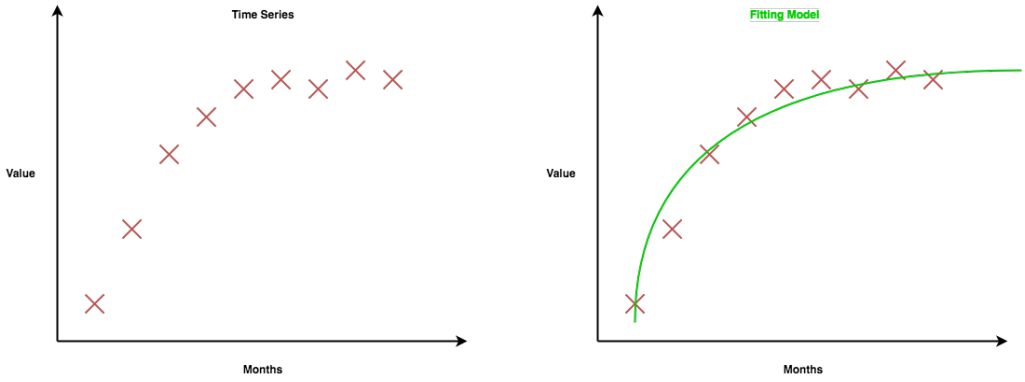

Figure 2.4: Example of a fitting Predictive Time Series Function (right graph).

2.9 Artificial Neural Networks

Artificial Neural Networks (ANNs) is a sub-field of Machine Learning that **investigates how simple models of biological brains can be used to solve difficult computational tasks**. The goal is not to create realistic models of the brain, but instead to develop robust algorithms and data structures that, by mimicking how brain cells interact with each other, can be used to model difficult problems.

ANNs are often considered the cutting edge of Machine Learning algorithms. The power of neural networks comes from their ability to learn the representation in training data and how to best relate it to the output variable that one wants to predict - as if they were mapping the data. Mathematically, **they are capable of learning any mapping function and have been proven to be a universal approximation algorithm**. The predictive capability of neural networks comes from their multi-layered structure of the networks. The neural network's data structure can learn to represent features at different scales or resolutions and combine them into more complex features [3].

Simply put, ANNs are a series of Neurons setup in layers, which connected to each other make a network. By adding up the process power of each neuron, these networks have been known to be able to, do tremendous computational work:

- Recognize objects in pictures;
- Understand handwriting;
- Transcribe speech;
- Make forecasts;
- Win games against the best players in the world.

2.10 Neurons

A neuron is the most basic unit and building block of an ANN. They are simple computational units that have weighed input signals and produce an output signal using an activation function.

Basically, the neuron is nothing more than a set of inputs, a set of weights, and an activation function.

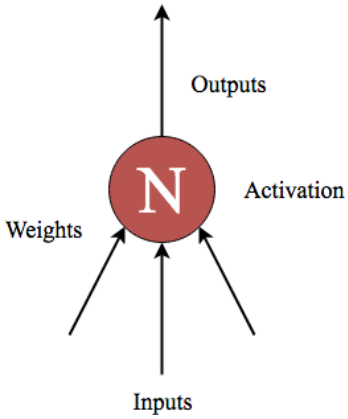

Figure 2.5: Model of a Neuron.

- **Inputs:** can either be the "raw" input features — say temperature, precipitation, and wind speed for a weather model — or the output of neurons from an earlier layer;
- **Weights:** are values that represent the **importance** given to each input;
- **Activation:** is the internal function of the neuron which, after all the inputs have been weighed, decides if the neuron is going to be activated or not;
- **Outputs:** the output of the neuron which can then go to another neuron or can be read.

2.10.1 Weights

When a neuron receives a series of inputs it applies **weights** to the inputs and sums them together. Weights are a representation of how *important* an input should be and how much should it affect the neurons decision. The way that inputs and weights are connected can be represented as follows:

$$Y = \sum (weight * input) + bias$$

The **bias** is an optional constant that is added as good practice for scaling purposes. The value of Y, which is a value that will be used by the activation function, can be anything ranging from $-\infty$ to $+\infty$. The neuron really doesn't know the bounds of the value.

2.10.2 Activation Function

Since the neuron does not know the bounds of the Y value, it uses an activation function which, given a threshold, **decides if the neuron is going to be activated or not**. The activation of the neuron affects other neurons and consequently impacts the decisions made by the network.

There are a multitude of different activation functions, each with its own purpose, but for the scope of this thesis we found that it should only be necessary to understand the general architecture of a neuron [39] [11].

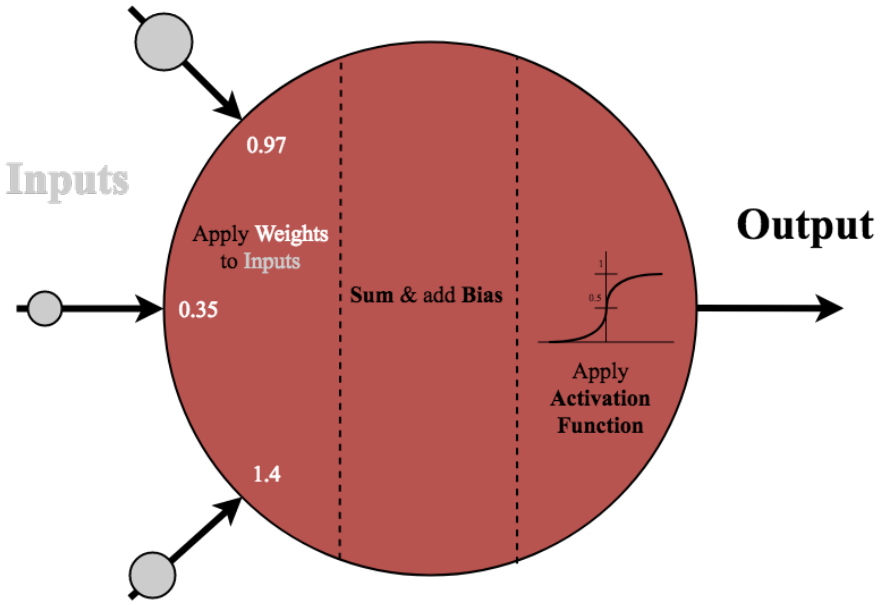

Figure 2.6: Model of how an Artificial Neuron handles input.

2.11 Layers

Regular ANNs are made up of multiple layers. A layer **represents a row of neurons** and each neuron from one layer is connected to at least one neuron from another layer, but never to adjacent neurons.

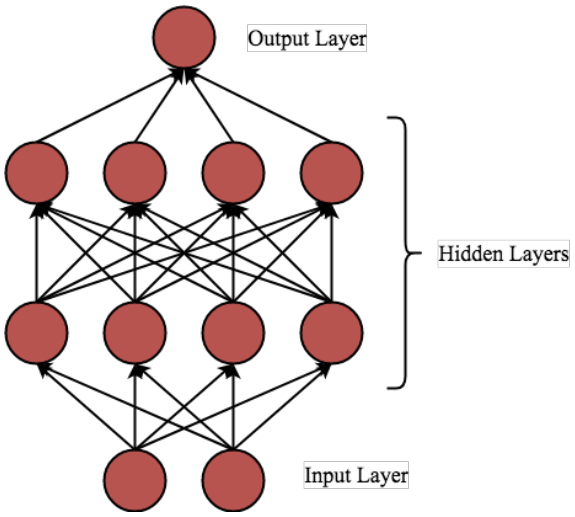

Figure 2.7: Model of an Artificial Neural Network's topology.

2.11.1 Input Layer

The layer that takes the input from a dataset is called the **input layer or visible layer**. This is the exposed layer of the network. Often this layer is made up of a single neuron for each column in the dataset - these are not neurons per se, they **just pass the input onto the next layer** of the network.

2.11.2 Hidden Layers

The layers after the input layer are called hidden layers because they are not directly exposed to the input. There can be multiple hidden layers. These layers are **the ones that makeup the whole of the processing power that exists in ANNs**, i.e. the more hidden layers there are, the more in depth the learning process of these networks becomes - that is why the field that is concerned with studying very large neural ANNs is called *Deep Learning*.

2.11.3 Output Layer

The final layer of a network is called output layer and it is responsible for **outputting a value or vector of values that correspond to the format required for the problem**. There can exist specific problems which require an output layer to have multiple neurons. For example, when a neural network is tasked with determining if the animal in a picture is a cat, a dog or a bird - each neuron in the output layer can return the probability of the animal in the picture being one of those three animals respectively.

2.12 Training Artificial Neural Networks

2.12.1 Stochastic Gradient Descent

The classical training algorithm for neural networks is called **stochastic gradient descent**. This training method goes as follows:

1. one row of data from the training data is fed to the network as input - again, each column of the row is fed to a single neuron in the input layer;
2. the values of the input go along the network and start activating different neurons as they go through the layers;
3. the network then returns an output - the process described in step 2. and 3. is called a ***forward pass*** on the network;
4. the output of the network is compared to the expected output and an *error* is calculated.
5. This error is then propagated back through the network, one layer at a time, and the *weights are updated* according to the amount that they contributed to the error - this process is called **backpropagation**;

6. the process is repeated from step 1. for all the rows in the training data.

The training data is almost always run through the network more than once. One round of updating the network for the entire training dataset is called an ***epoch***. A network may be trained for tens, hundreds or many thousands of epochs.

2.12.2 Weight Updates

But there is a problem in updating the weights through backpropagation in every epoch.

Since only one row is being fed on each forward pass until we complete an epoch, each weight update is going to be a reflection of the processing of multiple rows passed individually and thus making a lot of changes in weights. It means that the network is not being very efficient at associating things learned in each epoch because ***multiple rows are never looked at as a whole****. This can result in fast but very chaotic changes to the network.*

The solution for this problem is **batch learning**. By bundling up a number of rows into a batch before updating the weights, the network is able to think more broadly and encompass more knowledge before each "conclusion".

This of course, is more computationally taxing and so the *batch size* is often reduced to a small number, such as tens or hundreds of examples.

2.13 Recurrent Neural Networks

Imagine the following scenario: we want to make a weather forecast using a regular feed forward network, i.e. network that just passes its input and results from one layer to the next.

We can create a network and set it up with a batch size of 5 days. As an epoch progresses, the network will learn to understand multiple individual relationships that span for 5 days. This may work but because we are only focusing on a 5 day span at a time, we are neglecting other acknowledgements made in previous days. We have found a problem here:

Regular Artificial Neural Networks have no notion of order in time. They look at a grouping of observations but cannot relate them with previous knowledge. So we can only make very limited predictions in time series problems.

Recurrent Neural Networks were created with this exact problem in mind.

2.13.1 Architecture

Recurrent Neural Networks or RNNs are a **special type of neural network designed to accurately address sequence problems**. They understand past behaviors by storing knowledge in a context storage and then feeding that knowledge back to the network as input. This can be seen

as an architectural loop. When going through an epoch, RNNs undergo the following steps as described and illustrated below on figure 2.8:

1. initially the first row of input i_1 goes through the input layer;

2. the input layer then gives the input to the hidden layers;

3. after doing a forward pass through all the hidden layers, the knowledge from that forward pass k_n is stored in a *context storage*;

4. the input layer gives another input i_n to the hidden layers along with the previously stored knowledge k_{n-1};

5. steps 3. and 4. are repeated for each training row iteration n until an epoch has been completed.

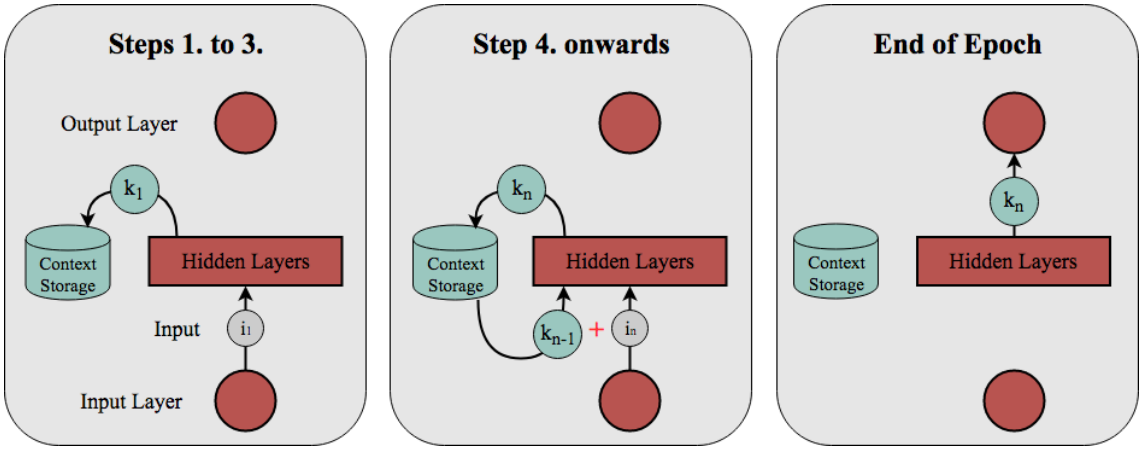

Figure 2.8: Steps of an epoch through a Recurrent Neural Network.

Since the recurrent connections add state or memory to the network, it is able to learn broader abstractions from the input sequences. RNNs also ***enable neurons to loop through themselves and connect to adjacent neurons, there can be no concept of "layer" in RNNs***.

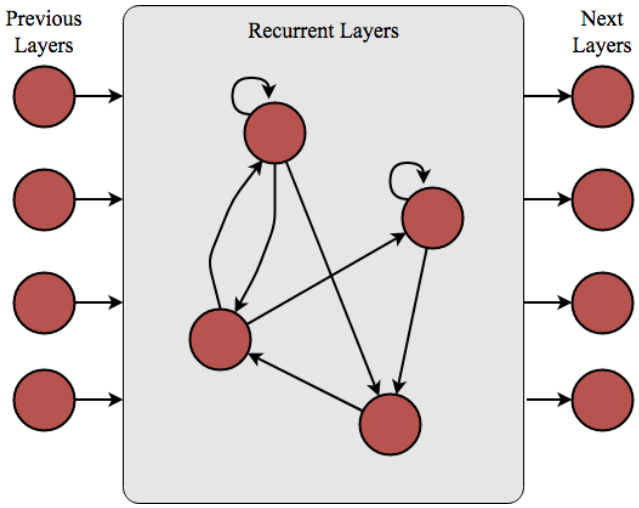

Figure 2.9: An axample of the recurrent behaviour of neurons in RNNs.

But for the techniques to be effective on real problems, two major issues needed to be resolved [3, 37, 22, 7].

1. ***How do we train the network using backpropagation?***
2. ***When training, how do we stop gradients from vanishing or exploding from very deep networks?***

2.13.2 Training Recurrent Neural Networks

The standard technique for training regular feedforward neural networks is to backpropagate error and update the network weights. Backpropagation does not work in a recurrent neural network, because of the recurrent or loop connections that may exist in neurons. This problem was addressed with the creation of a new Backpropagation technique called **Backpropagation Through Time** or BPTT.

Instead of performing Backpropagation on the recurrent network as stated, BPTT **unrolls the structure of the network and creates copies of the neurons that have recurrent connections**. For example a single neuron with a connection to itself (A $\rightarrow$ A) is represented as two neurons with the same *weight values* (A $\rightarrow$ B). This allows the cyclic graph of a recurrent neural network to be turned into an acyclic graph like a classic feedforward neural network, and then Backpropagation can be applied.

2.13.3 Achieving Stable Gradients During Training

When Backpropagation is used in **very deep neural networks** and in unrolled recurrent neural networks, there can be some instability in the adjustment of the weights. As we could see in Figure 2.8 the loops that go on in the inner layers of the network keep on adding new k knowledge as they go. At each new loop n the knowledge k_n can be formulated as:

$$k_n = forward\_pass(k_{n-1} + i_n)$$

Since this is a recursive process, each k_n contains all previously extracted knowledge which has been run through the network:

$$k_n = k_1 + ... + k_{n-1}$$

When the network is unfolded for backpropagation through time, **the gradients that are calculated in order to update the weights, can become very large numbers called *exploding gradients* or very small numbers called the *vanishing gradient* problem**. These large numbers in turn are used to update the weights in the network, making training unstable and the network unreliable. In recurrent neural network architectures, this problem has been alleviated using a new type of architecture called the **Long Short-Term Memory Networks** that allows deep recurrent networks to be trained.

2.13.4 Long Short-Term Memory Network

Long Short Term Memory networks or LSTMs are a special kind of RNN, **capable of learning long-term dependencies** thus overcoming the gradient explosion and vanishing. They were introduced by Hochreiter & Schmidhuber in 1997, and have been refined and popularized by many other authors [36]. They work tremendously well on a large variety of problems, and are now widely used. Instead of neurons, LSTM networks have *memory blocks* that are connected into layers. A block has components that make it smarter than a classical neuron and it also has a memory for storing recent sequences [32].

This architecture enables LSTMs them to create very large recurrent networks that in turn can be used to address difficult sequence problems in machine learning and achieve state-of-the-art results [3].

Chapter 3

State-Of-The-Art

Contents

Now that we have set a solid conceptual background, we can search for existing work that have tried to solve the problems stated on chapter 1. By doing so, we can learn more about new design and implementation ideas for our solution proposal, and most importantly, we can **validate our solution proposal**.

By the end of this chapter, we will have a good grasp of some concepts and methods that can allow us to approach our problems and thus we can formulate a clear problem statement from which we will develop our solution.

3.1 Introduction

The purpose of this thesis lies on the solution of two problems:

Q.1.: Could we use computer science to help us take Time Series Analysis and behavioral pattern recognition a step further?

Q.2.: ***How is the software application industry using behavioral data and Time Series Analysis to help us learn? & Can the approach used be improved upon in any way?***

3.2 Competing Solutions For Question #1

As we will see in chapters 4 and 5, a conclusion will be reached that we will be creating a Long Short-Term Memory network to make predictions based off data. And then, we will attempt to

extract the knowledge that the network had acquired when analyzing the data. This information helps us to prune any complementary information and focus on a more specific aspect:

How are scientists extracting knowledge acquired from artificial neural networks?

Artificial neural networks are not new, they have been around since the early 1940s [27], but there has only been a scarce amount of publicly known research development on the field of ANN analysis; particularly regarding methods which enable the extraction of knowledge from ANNs. Two of the most well known articles are from 2002 [23] and 2004 [9] by Olden et al.

Julian D. Olden, co-author of both of these articles, is an ecologist concerned with understanding how the statistical analysis made by artificial neural networks could be used for gaining knowledge of the causal relationships driving ecological phenomena. In both of these articles the authors discuss and compare a series of methods which allow for the **quantification of variable importance in artificial neural networks**. On the first article they create a variable importance quantification method using connection weights. And on the second article, they compare nine variable importance quantification methods - including their own - and rate them according to how well each method classifies the importance of each variable, based on a comparison made with simulated data whose properties and patterns were known.

According to their results the two best performing methods were:

1. **Connection Weight Method**, with 92% accuracy;
2. **Sensitivity Analysis**, with around 70% accuracy;

3.2.1 Connection Weight Method

The connection weight method or randomization test for input-hidden-output connection weight selection in neural networks, is the method created by Olden et al. in the initial article published in 2002 [23]. The method improves upon common practices implemented in sensitivity analysis and in Garson's algorithm - another method used for determining relative importance of each input variable in the network. The randomization protocol works as follows:

1. construct a number of neural networks using the original data with different initial random weights;
2. select the neural network with the best predictive performance, record initial random connection weights used in constructing this network, and calculate and record:
   (a) input-hidden-output connection weights: the product of input-hidden and hidden-output connection weights for each input and hidden neuron;
   (b) overall connection weight: the sum of the input-hidden-output connection weights for each input variable;
   (c) relative importance (%) for each input variable based on Garson's algorithm;

3. randomly permute the original response variable ($yrandom$);

4. construct a neural network using *yrandom* and the initial random connection weights;

5. repeat steps (3) and (4) a large number of times (e.g. 999 times according to the original study) each time recording 2(a), (b) and (c).

The connection weight method has shown great results, with an average accuracy of 92% [9]. The only problem is that the documentation supplied by the author suggests that this method has only been tested and implemented in regular feed forward artificial neural networks. Models for networks with more complex architectures, such as recursive neural networks, have not yet been documented.

3.2.2 Sensitivity Analysis

Sensitivity analysis is a conglomerate of multiple different methods which focus on varying each input variable while holding all other input variables constant, as if the neural network was a *black box* that cannot be manipulated. The goal of these methods is to determine how much the predictive performance of the neural network is affected by the variations in the values of the input variables. Once all input variables have been tested, the overall importance of each is assessed based on the perturbation that the variation of the variable caused to the network's predictive performance. The most commonly used methods for sensitivity analysis are:

- **Traditional Sensitivity Analysis:** varying each input variable across its entire range;
- **Partial Derivatives:** create infinitesimal changes in the input variable
- **Input Perturbation:** adding a small amount of white noise to each input variable (around 50% according to the authors [12])

All of these methods are equally accurate to some extent, though they may vary slightly, their average accuracy when evaluating variable importance is of 70% [9]. The biggest drawback with sensitivity analysis is that it requires a significant amount of resources since neural networks are required to be ran multiple times for each variable variation.

3.3 Competing Solutions For Question #2

Question 2 is concerned with reducing and possibly fully avoiding the afflictions motivated by the implementation of the one-size-fits-all paradigm in tracking applications (section 1.3 of chapter 1). Therefore when looking at the state-of-the-art for this solution, we should look at existing tracking applications and how they were designed with a view to attack these problems.

3.3.1 MyFitnessPal - Calorie Counter Application

MyFitnessPal is a calorie counter application which provides users with an extensive food product database for them to keep track of their diet. It enables users to keep track of their dietary activity and goals and provides them with relevant information to help reach these goals. Being **body weight** the app's main traceable fitness goal. Since it was considered one of the best calorie counter applications of 2016 [35], MyFitnessPal has grown to become not only a calorie counter application but a social fitness network where users can participate in in-app forums and learn from fitness articles posted by the MyFitnessPal blog team.

3.3.1.1 Problem Approach

In order for the application to provide dietary guidelines it first creates a profile of the user. This profile then determines the recommended amount of ingested calories in order to achieve a set goal. The majority of the parameters which the user has to input in order to begin tracking their goals have an influence in calorical ingestion recommendations, as well as micro and macro nutrient ingestion recommendations [17]. But the main parameters which directly influence the dietary guidelines provided by the application are:

- **Birthday**;
- **Gender**;
- **Height**;
- **Current Weight**;
- **Goal Weight**: determines whether the user wants to gain, lose or maintain weight;
- **Weekly Goal**: determines the rate at which the user wants to achieve their goal: having seven possible answers (the following options use Kilograms as the unit for mass):
  - Lose 1 kg per week;
  - Lose 0.75 kg per week;
  - Lose 0.5 kg per week;
  - Lose 0.25 kg per week;
  - Maintain my current weight;
  - Gain 0.25 kg per week;
  - Gain 0.5 kg per week;
- **Activity Level**: this paramenter represents the user's average activity level, disregarding any type of *intentional workout sessions* - and has the following possible answers:
  - **Not Very Active**: spend most of the day sitting (e.g. bank teller, desk job);

- **Lightly Active**: spend a good part of the day on your feet (e.g. teacher, salesperson);
- **Active**: spend a good part of the day doing some physical activity (e.g. food server, postal carrier);
- **Very Active**: spend most of the day doing heavy physical activity (e.g. bike messenger, carpenter)

Once these parameters are set, the user is then able to track how many calories they should ingest in order to achieve their goal. But since users can also engage in intentional workout sessions, the app also enables them to input cardiovascular activities, which then influence calorical ingestion recommendations. Not only that, but as a way to further improve the efficiency of the guidelines provided by the application, it can use third party integrations to respond to a user's specific daily activity level. Some of these third party applications monitor how much calories they burn over the course of the day which then updates their MyFitnessPal calorie goals based on this information [16].

3.3.1.2 Approach Appraisal

This application does a great job at narrowing down the users categorization by asking for fairly relevant information, which helps fit each user in a relatively unique category. And the usage of third party applications is a great method for dynamically adjusting the provided guidelines and dietary recommendations. But as was reported by the MyFitnessPal team:

> *"While the calorie goals we calculate for you are based on statistical averages, our millions of users have demonstrated these goals are accurate enough to provide positive results for almost anyone."*

The **guidelines which they provide are based on statistical averages** and therefore do not cater and respond specifically to each users needs. Even when using third party applications, these can provide heart rate and other measurements, but they do not take into account whether or not each user's results are matching the applications guidelines and therefore do not make an effort to dynamically determine which dietary options taken by the user are positively influencing their goals [16, 17].

In summary, despite doing a great effort, the application **does not truly avoid the one-size-fits-all paradigm** - it simply narrows down the population in which an individual is inserted.

3.3.2 Noom Coach - Health & Lifestyle Coach

Noom Coach is the core application of a Health & Fitness application bundle by Noom Incorporated. Noom Inc.'s goal is to help users attain a healthier lifestyle by providing them with expert advice and analysis to keep them on track of their goals. This expert advice is provided by health care specialists who work to make each plan unique and tailor-made for each user. They are professionally trained to help users become more self-aware of healthy and harmful habits, guide them into these new habits and sustain them.

Simply put, Noom Coach is an application which provides personalized coaching by giving users traceable dietary, fitness and lifestyle goals uniquely adjusted by health care professionals. In terms of nutritional data, the application is meant to help users with meal tracking. But it only takes into account the calorical value of each food product ingested, not providing any information regarding macro and micro nutrient ingestion [20].

3.3.2.1 Problem Approach

As the majority of health & fitness applications, in order for it to provide relevant guidelines, the application firstly prompts users to input personal data. Being a coaching application, Noom Coach asks a series of questions that go beyond the standard weight, height and age parameters, and instead goes deeper into the lifestyle of the individual by asking questions on a more personal level. When prompting the app to start a course, it asks the following questions with the following answers listed beneath each question:

1. **Q:** When it comes to snacking, do you like salty or sweet?
   (a) Salty
   (b) Sweet
   (c) Both
   (d) Other
2. **Q:** Why do you want to start getting healthy?
   (a) More energy for my day-to-day
   (b) Be able to keep up with my kids
   (c) More confidence
   (d) Avoid a bad health situation in the future
   (e) Accomplish a physically demanding goal (run a race, go backpacking, travel, etc.)
   (f) Other
3. **Q:** What are your favorite activities?
   (a) Walking
   (b) Running
   (c) Playing with my kids
   (d) Hiking
   (e) Reading
   (f) Gardening & Yardwork
   (g) Team sports

(h) Cooking & Baking

(i) Biking

(j) Gym

(k) Racquet sports

(l) Golf

(m) Exercise classes

(n) Swimming

(o) Extreme sports

(p) Dancing

(q) Yoga

(r) Games

(s) Skiing & Snowboarding

(t) Ice sports

4. **Q:** What are you looking for in a coach?

   (a) Accountability

   (b) Continuous support

   (c) Motivation

   (d) Tough but honest feedback

   (e) Guidance

   (f) Close monitoring

   (g) Other

5. **Q:** What are the obstacles keeping you from getting healthier?

   (a) Not enough walking

   (b) Snacking on junk food

   (c) Overeating at night

   (d) Stuck at a plateau

   (e) Going overboard on weekends

   (f) Not enough physical activity

   (g) Don't love veggies

   (h) Always feeling hungry

   (i) Sweet tooth

   (j) Stress and a busy lifestyle

(k) Confusion about what to eat

(l) Social eating situations

(m) Other

Besides understanding the user's tastes, as well as fitness and lifestyle goals - as made evident by the Australian Government's Department of Health's National Eating Disorders Collaboration - some of these questions also help trace a psychological profile for possible eating disorders that the user may have [31].

After these questions have been answered, the Noom Coach application asks the user to enroll in one of three possible programs:

- **Healthy Weight Program**: for individuals who are trying to lose weight in a healthy way, and what to put a definite end to those bad eating and fitness habits;
- **Diabetes Prevention Program**: for individuals who are trying to lower their blood sugar levels and reduce their risk of type 2 diabetes;
- **Hypertension Prevention Program**: for individuals who are trying to lower their blood pressure and reduce their risk of hypertension;

These are all paid programs and only after subscribing to them does the user get access to a personalized plan, personal specialist and a community of other users who can help and support each other with their goals.

3.3.2.2 Approach Appraisal

Of both of the mentioned applications, this one truly finds a solution to our problem: **by using a trained specialist, one is catering specifically to each individual user's needs**. Note that this also relies on the quality of the work executed by the specialist, but given that it has all of the potential to truly solve our issue, we can confirm that it can indeed solve the problem at hand.

But from a software engineering point of view, the problem is not completely solved in terms of optimization, i.e. **the solution is effective, but not efficient**. This happens because the specialists are human beings, therefore they can only answer each user's questions and needs individually and one at a time. By transferring the analytical knowledge used by the specialists when diagnosing and trying to understand the user's needs into a purely software and mathematical approach, we can increase the efficiency of this job.

3.4 Summary

Competing solutions for Question 1 have shown us that there exist several software methods for extracting behavioral patterns from data. By joining these methods with artificial neural networks that learn from data, we can extract behavioral patterns from the networks.

Knowing this, we can foresee that the creation of a more complex method based on the methods created, which would extract simple behavioral patterns from neural networks, could help push time series analysis and behavioral pattern recognition a step further.

When searching for competing solutions for Question #2, we have found that despite there being good efforts in solving the problem, these solutions are not as efficient as they could be since they imply human intervention. By creating a purely software and mathematical method that would mimic the analytical knowledge provided by human professionals, we could create an unbiased approach that would replace the implementation of the one-size-fits-all paradigm.

In sum, **both problems have been validated** and we can now start the development the problem statement.

Chapter 4

Problem Statement

Contents

Now that we have validated our initial problems, we can finally create our problem statement and create a solution proposal.

4.1 Reviewing The Issues

We will start by revisiting the issues initially enounced in chapter 1 and understanding their correlation. We will summarize what the problems are and how they can be decomposed and then, in the next section, we will take a deeper look at the correlation's specifics. So, let us start by attending question number 1, raised on section 1.2:

Q.1.: Could we use computer science to help us take Time Series Analysis and behavioral pattern recognition a step further?

First we need to clarify any technical language and then reformulate the question so that we can reach a compelling answer. When talking about "using computer science to take TSA and behavioral pattern recognition a step further" we are referring to upgrading or replacing traditional TSA methods which rely heavily on the manual application of statistical methods to recognize patterns in data. There exist a series of terms in this definition that overlap, which can be simplified. According to what we know about time series forecasting and time series analysis, let us rethink this question with another perspective in mind:

1. **Time Series Analysis** is the field **concerned with detecting behavioral patterns** by attempting to comprehend how time-series values change, at what rate, and how they can be replicated;

2. There are **Machine Learning algorithms** concerned with Time Series Forecasting, which **search for behavioral patterns** in time-series datasets and make predictions based on these patterns;

3. Since these **Machine Learning algorithms** are made to detect behavioral patterns before making any type of prediction - we can say that they **perform Time Series Analysis before making a Time Series Forecast**.

We can now see that there is a *deep connection* between Time Series Forecasting and Time Series Analysis in Machine Learning algorithms. Meaning that we can reformulate the question and make it more objective by instead asking:

NEW Q.1.: Can we extract the behavioral patterns detected by Time Series Forecasting algorithms?

Now that we have reached a more clear version of Question 1, we can move to the next question - which is compound by nature, but will be looked at as two separate questions as they were initially constructed.

Q.2.1.: How is the software application industry using behavioral data and Time Series Analysis to help us learn?

We have learned through research that in the software application industry the most evident product that is using behavioral data are applications called **tracking applications** which **let users record their behaviors and get tips on how to better reach their goals**, as if the applications themselves where *software coaches*. These applications usually do so in the following manner:

1. Ask users a series of personal questions;

2. Use the **one-size-fits-all paradigm** to label their users, using broad and sometimes ambiguous labels;

3. Advise each user to follow a specific approach that is best-suited for the majority of identically labeled individuals.

By looking at the one-size-fits-all paradigm we asserted that there are three possibly emergent **problems for people that do not adequately fit into the categories** which they were assigned to:

1. Over-extended time frame for expected results;

2. Unreachable results;

3. Unexpected neglect or stress on overlooked details.

These problems hinder user experience, but most importantly, they can **lead to physical harm** for users of health and well-being tracking applications. Leading us to the second part of the question:

Q.2.2.: Can the approach used be improved upon in any way?

By understanding the answer to question 2.1. and understanding that the implementations of the One-Size-Fits-All paradigm is the root of the problems found in most tracking applications - we can now reformulate this question to address this specific problem, thus removing any ambiguity and making a new final question that is more extensive but also more objective.

NEW Q.2.: Can we find an alternative for the implementation of the One-Size-Fits-All paradigm in tracking applications in order to remove their rooted bias?

4.2 Converging Both Issues

The questions have become more coherent and so the solution to the problem is becoming more apparent; but we still have to understand how these two problems relate, before making any progress towards developing a solution.

And so, we shall dive into the correlation between Q.1. and Q.2.. As means of clarification, this search for the correlation between them, will help us with redundancies and overlaps found when searching for the answers to both of these question. Since both questions are referent to the topic of *behavioral pattern assessment*, it is only natural that **some of the answers to these questions will be very similar** if not identical in nature. Therefore these overlaps will help us **create a solution which targets both problems**. Let us have a closer look at both questions, now paired together:

Q.1.: Can we extract the behavioral patterns detected in Time Series Forecasting algorithms?

Q.2.: Can we find an alternative for the implementation of the One-Size-Fits-All paradigm in tracking applications in order to remove their rooted bias?

Considering that Q.2. is more specific than Q.1. - i.e. it is concerned with the solution for a software implementation problem instead of being perceived as a broader question with a lot more abstract and *higher-level* solutions - we will try to answer Q.2. first.

As was described in section 1.3, the One-Size-Fits-All paradigm happens when an heterogeneous group of subjects are identically labeled, and then treated as equal. In tracking applications, this means that each identically labeled subject will be given a treatment that is most appropriate to a "model subject" and not to that individual's needs.

A possible **solution** seems to be to have **each individual treated as a unique subject**, without having any external bias labeling them as someone whom they are not 100% identical to.

How can we identify unique individuals in tracking applications?

Since tracking applications are concerned with tracking each users behavior, we can **use their recorded behavior to identify them** and determine what is working best for them. Considering that these tracking applications were created with the intent of helping users determine what behavior helps them reach their goals more efficiently, we could ***try to understand what behaviors are affecting these goals and at what rate***. If we go back to section 1.2 on the definition of TSA we find that it clearly resembles this last statement:

> *"As the name implies, Time Series Analysis is when we look at a time-series dataset and try to comprehend how values are changing, at what rate are they changing, and how they can be replicated."*

This opens up a new way of solving our problem. A possible solution for the One-Size-Fits-All paradigm could be the ***creation of a method which tries to understand how and what behaviors are affecting a user's goals, so that the tips provided by the application are given according to this understanding of which behaviors are working best for the user's progress towards these goals***. The similarity between this definition and the definition of TSA points us towards reaching the conclusion that we can use TSA to understand how and at what rate are users' behaviors affecting their own goals.

And with the same thought process put in the deconstruction of the initial version of Q.1. (from section 4.1) we can state that:

1. **TSA** is concerned with **searching for behavioral patterns**;
2. Some **Machine Learning algorithms that can do TSF**, do so by first **searching for behavioral patterns**;
3. Therefore, these **Machine Learning algorithms perform TSA**;
4. Therefore we can say that TSF Machine Learning algorithms can understand how and at what rate are application user behaviors affecting their own goals.

Curiously enough, this deconstruction has lead us to the same question: *in order to understand how and at what rate are application user behaviors affecting their own goals...*

Q.1.: Can we extract the behavioral patterns detected in Time Series Forecasting algorithms?

Even though when analyzed individually these questions may differ in scope, both Q.1. and Q.2. can be boiled down to the same exact question. And as we have seen in section 3.2 of the previous chapter, we do have methods for the extraction of knowledge from artificial neural networks. By using these methods and applying them to time series forecasting we should be able to create a behavioral patterns extraction method for time series forecasting algorithms.

4.3 Main Goal

Given this analysis, we have now managed to better understand the scope of the problem and the main issues that we are attempting to solve. By going back to the introductory chapter 1, and revisiting the main motivations of this work, we can define our main goal as being:

> ***The creation of methods that extract the behavioral patterns detected by state-of-the-art time series forecasting algorithms and present them in an easy to understand format.***

This method will hopefully serve as a solution to:

INITIAL Q.1.: Could we use computer science to help us take Time Series Analysis and behavioral pattern recognition a step further?

ANSWER: Yes. By using computer science to create a method which can extract new behavioral patterns we can help take Time Series Analysis and behavioral pattern recognition a step further, by being able to use state-of-the-art forecasting algorithms to help us learn new concepts from previously thought to be overly large and hard to assimilate portions of data.

Q.2.: Can we find an alternative for the implementation of the One-Size-Fits-All paradigm in tracking applications in order to remove their rooted bias?

ANSWER: Yes. By creating a method that can extract a user's behavioral patterns, we can determine which behaviors are helping the user make more progress towards their goals. This way, each user is evaluated individually and the tips given to them will be specific to what is working for them without any bias.

4.4 Solution Proposal

We will be dividing our implementation work into three parts, split by three chapters:

1. **Chapter 5 - Implementation Foundation:** Discussion of which machine learning algorithms, datasets and technology to use for the implementation of our solutions;
2. **Chapter 6 - Forecasting Algorithm:** Creating a time series forecasting algorithm using a Long Short-Term Memory network (as concluded from the previous chapter);
3. **Chapter 7 - Behavioral Pattern Extraction Methods:** Creating behavioral pattern extraction methods which will extract the patterns detected by the network using different forms of sensitivity analysis.

As will be discussed on chapter 5, the forecasting algorithm implemented will be forecasting the pollution of the atmosphere in Beijing during the peak of the winter of 2010, based on a portion of a dataset which contains hourly data regarding the concentration of PM2.5 particles in Beijing from January 1st, 2010 to December 31st, 2014.

Initially the goal was to use time series data with records of human behavior for continuity purposes, since one of the initially raised issues was related to the tracking of daily human behaviors through tracking applications. Unfortunately, we could not find a good time series dataset containing logs of human behavior because the datasets found had no scientific background and so, we would not be able to validate the accuracy of the behavioral patterns extracted with our method. Therefore, we decided to choose a time series dataset that contained other types of scientifically studied behaviors, in this case: the behavior of atmospheric particles and weather conditions.

After creating and tuning the Long Short-Term Memory Network we will create two behavioral pattern extraction methods which will detect two simple behavioral patterns using sensitivity analysis:

1. **Behavior Connection Weight Pattern:** how much the changes in a behavior induce change in another behavior, i.e. if a change in behavior A is related with a small or a big change in behavior B;

2. **Average Linear Correlation Direction Pattern:** how the variation in a behavior is correlated with the variations of another behavior, i.e. if an increase or decrease in behavior A is related with an increase or decrease in behavior B;

By the end of the implementation, we will have created **two simple methods that combined, will enable us to understand how much each behavior is influencing a target behavior and how changes in each behavior are related to changes in the target behavior** - respectively, how much each meteorological condition is affecting the pm2.5 concentration in Beijing and how the variation of these meteorological condition are affecting the variations of concentration of pm2.5.

All further implementation decisions will be covered in detail in their respective chapters.

4.5 Evaluation Strategy

Given the scientific and research nature of our work, the successful implementation of our solution will be based on the **following requirements**:

- **Requirement 1:** the root-mean-square error of the predictive performance of the neural network must be lower than 15% of the range of the predicted attribute's values in the training data;

- **Requirement 2:** the methods were able to extract behavioral patterns detected by the Time Series Forecasting algorithms;

- **Requirement 3:** the methods were at least as accurate at extracting correct behavioral patterns as the underlying methods from which they were based on;

- **Requirement 4:** the behavioral patterns extracted were unbiased and unique to the dataset from which they were collected.

Each of these requirements will be deemed *accomplished* given the following conditions:

4.5.1 Meeting Requirement 1

The root-mean-square error of the predictive performance of the artificial neural network must be lower than 15% of the range of the predicted attribute's values in the training data. Calculated like so:

$$maxRMSE = (maxTrainingValue - minTrainingValue) * 0.15$$

For example, if the values of the predicted attribute in the training data, range from 1500 to 2500, than the minimum root-mean-square error of the prediction must be 150, given that:

$$150 = (2500 - 1500) * 0.15$$

4.5.2 Meeting Requirement 2

It was possible to create a method which enabled the extraction of behavioral patterns from the patterns assessed by a Time Series Forecasting algorithm.

4.5.3 Meeting Requirement 3

The percentage of extracted patterns which could be proven to exist, must at least be the same expected percentage of extracted patterns that could be proven to exist using the methods from which the developed method was based on.

In order to prove that a behavioral pattern does exist, there must be relevant scientific and statistical research whose results comply with the behavioral patterns detected.

4.5.4 Meeting Requirement 4

The method developed must not make any assumptions on the data, using simply mathematical and statistical methods which do not manipulate or alter the values of the data used.

4.6 Research Questions

For the correct development of our solution, the following questions were raised and answered to clarify how to deal with possible coming obstacles.

4.6.1 How to optimize a neural network if its baseline performance does not meet Requirement 1?

If the baseline performance of the developed neural network does not meet Requirement 1, then the network's training method and topology must be optimized through each step of the following methods, until Requirement 1 is met :

- **Training Optimization Methods:**

1. Readjust the **number of epochs** by:
   - setting an initially low number of 50 epochs and testing the network's performance;
   - increasing the number of epochs by twofold (2x the previous number) and testing the network's performance;
   - repeating the previous step until no substantial improvement in performance is seen for the exponential increase in number of epochs.
2. Readjust the **batch size**: if the batch size cannot be as large as the number of instances in the dataset, then the batch size must be maximized as much as possible without causing the algorithm to "blow up" and forcibly terminating its execution;

- **Topological Optimization Methods:**

1. Readjust the **number of hidden layers** by:
   - starting with just an input and an output layer and testing the network's performance;
   - adding one hidden layer at a time with the same amount of neurons as the input layer and testing the network's performance;
   - repeating the previous step until no substantial improvement in performance is seen for the increase in number of hidden layers.
2. Readjust the **number of neurons in the input layer** by:
   - adding one neuron to the input layer and testing the network's performance;
   - repeating the previous step until no substantial improvement in performance is seen for the increase in number of neurons in the input layer.
3. Readjust the **number of neurons in the hidden layers** by:
   - starting with just one neuron on the first hidden layer and testing the network's performance;
   - adding one more neuron to the layer and testing the network's performance;
   - repeating the previous step until no substantial improvement in performance is seen for the increase in number of neurons in the initial hidden layer;
   - repeat all of the previous steps for each hidden layer.

4.6.2 What to do if the percentage of correct extracted patterns does not meet Requirement 3?

If the percentage of correct extracted patterns is smaller than the expected percentage of extracted patterns that could be proven to exist using the methods from which the developed method was based on, then the methods used must be reevaluated and readjusted.

If when optimized, the underlying methods still do not meet requirement 2, then these must be replaced with more accurate methods.

4.7 Main Contributions

The work developed represents only a shallow dive into the waters of behavioral pattern extraction. The ability to extract new behavioral patterns from data could be immensely beneficial to many different areas and fields, namely any field that relies on discovery and expansion of new concepts. In the following subsections we will be discussing the most evident contributions in some of these fields.

4.7.1 Time Series Analysis Field

Using traditional mathematical methods in Time Series Analysis can be very cumbersome. The development of Machine Learning and Artificial Neural Networks has been pushing the benchmarks of predictive formulas to new boundaries. By implementing these new techniques on Time Series Analysis, we could help reduce the amount of work necessary to achieve good results and even change how the field will grow.

4.7.2 Traditional Scientific Field

Every traditional scientific field bases its development on the scientific method: a problem is raised, a hypothesis is formulated as a theoretical response to the problem and then experimentation takes place to prove or disprove the hypothesis. With new behavioral patterns being found from large amounts of new data, new hypothesis could then be formulated and tested to discover new phenomena.

4.7.3 Artificial Intelligence Field

Artificial Intelligence has been growing immensely in the past few decades. Increasingly more accurate and powerful Artificial Neural Networks have been on the forefront of predictive algorithms. But many research are concerned with the black box nature of these networks. The way that they process information is still an enigma. By being able to extract some of the patterns detected by these algorithms, we can begin to understand how these algorithms think and ultimately uncover new potential.

4.8 Summary

In this thesis we will be creating two methods that will be extracting the behavioral patterns detected by a Time Series Forecasting algorithm. A forecasting algorithm will be created to learn and make prediction of the concentration levels of PM2.5 particles in Beijing based on meteorological conditions, and then the methods will extract the patterns recognized by the algorithm.

The methods must successfully extract the patterns without manipulating the values of the dataset and obtain an average evaluation which is at least equal to the underlying methods from

which it is based on. In order to evaluate the performance of the methods, scientifically demonstrated patterns will be used to determine if the extracted patterns are valid.

The successful implementation of these methods will hopefully prove that more modern methods can be used for Time Series Analysis and that there is an alternative for the implementation of the One-Size-Fits-All paradigm in tracking applications, which can remove its rooted bias.

Chapter 5

Implementation Foundation

Contents

First we will start by looking for Time Series Forecasting Algorithms.

5.1 Choosing the Algorithm

In order for us to be able to extract any kind of behavioral pattern from an algorithm, the algorithm itself needs to be able to learn and assess these behavioral patterns.

As was mentioned several times before, Machine Learning is the field which is concerned with creating algorithms that can learn from data. Therefore, ML algorithms are the ones which we will be looking at. But as we know, there is a plethora of different algorithms, each with its category and only a handful of them are better suited for TSF. So we will start by going through each category of ML algorithms and choosing the ones that fit best.

5.1.1 Supervised or Unsupervised

We will start by looking at the learning styles. In this category we have:

- **Supervised learning:** algorithms that make predictions and then readjust their inner parameters by comparing their predictions with the actual values from the training data;
- **Unsupervised learning:** algorithms don't have any idea of what is and what is not correct and therefore just try to figure out data patterns by themselves.

Despite most TSF problems being addressed by Supervised ML algorithms, with these descriptions, it is hard to determine which one is a best fit - there are pros and cons to both:

- **Supervised learning**
  - **Pros:** by using training data, forecasts become more and more accurate with time and therefore, so do the behavioral patterns recognized by these algorithms;
  - **Cons:** training data can be too strict on the patterns found;
- **Unsupervised learning:**
  - **Pros:** since they are not restricted by any training data, they can find less common patterns;
  - **Cons:** too much flexibility may mean that the patterns recognized may never be sufficiently accurate.

So let us dive deeper and take a look at the inner groupings of each of these learning styles. In **supervised learning** we have classification and regression.

- **Classification:** when the output variable is a category such as Purchased or Sold or Spain or Italy;
- **Regression:** when the output variable is a real value, such as an amount of dollars or weight;

The patterns that we are looking to find are behavioral patterns that come from forecasting different behaviors. For example, forecasting if it is going to rain and how much it is going to rain according to temperature and humidity levels. By looking at both of these groups it seems that they can both be used for TSF, it just depends on the type of data that we are using.

Since we cannot conclude anything as of right now, we are going to look at the inner groupings of **unsupervised learning**.

- **Clustering:** discover the inherent groupings in data, such as grouping customers by purchasing behavior;
- **Association:** discover rules that describe large portions of your data, such as people that buy A also tend to buy B

Both of these groups lead to thinking that they can be good for our problem, since we are trying to detect behavioral patterns. But there is a key factor which is not taken into consideration when looking at clustering and association:

> *Both groupings analyze data that is not sequenced.* ***None of the algorithms of these categories take time into account when analyzing behavioral patterns****. The analyzed behaviors exist in an unknown instance of time.*

Therefore, these algorithms cannot detect behavioral patterns that change through time such as: when temperature increases, the amount of pollens in the atmosphere increases as well. And

that means a very strong restriction on the type of behavioral patterns found. Since time is a key factor of behavioral pattern recognition we can assert that **all unsupervised machine learning algorithms are unfit for Time Series Forecasting and subsequently behavioral pattern recognition**. We can then conclude that between these two learning styles we opt for **supervised machine learning algorithms**.

Also, since we are concerned with recognizing behavioral patterns, it is best if we use supervised algorithms that can train the models and **make sure that the forecasts are as accurate as possible before extracting any patterns** because this will influence their quality as well.

5.1.2 Parametric or Nonparametric

Now that we have determined that we will be using supervised machine learning algorithms, we can go further along this branch and look at the parameterization of these algorithms. In subsection 2.6 of chapter 2 we discussed parametric and nonparametric algorithms.

The existence or non-existence of parameters mostly refers to the assumptions that the algorithm makes on the underlying mapping functions of the data. This greatly influences the balance between flexibility and power of the predictions made.

- **Parametric:** the estimations made are restricted by a previously determined form - this means that if we say that the underlying function has to fit into a linear form, the algorithm will always assume that the prediction has to be of a linear form.
- **Nonparametric:** the estimations made are not restricted in any way and so the mapping function is free to assume the best form possible on the training data.

In this category, the most fitting option is fairly obvious. Parametric algorithms are faster and simpler, but they make assumptions on the behavioral patterns that exist in the data and for that reason they are not a good option since we are trying to approximate our mapping functions to the underlying functions of the data - whatever those functions may be. With that being said **nonparametric machine learning algorithms** seem to be the most fitting of both of these categories.

5.1.3 Artificial Neural Networks

Up until this point we have determined that we are going to **implement a Time Series Forecasting algorithm** so that we can then extract the behavioral patterns recognized from this data. We have also assessed that we are looking for a **supervised nonparametric machine learning algorithm**.

Based on common mentioning found in most of the literature referenced throughout this thesis we have found that there are 3 types of algorithms that fit this description:

1. **Naïve Bayes**
2. **Support Vector Machines**
3. **Artificial Neural Networks**

Another important key aspect is that the Time Series Forecasting problems that we are attempting to address are **Multivariate Times Series Forecasting** problems - since we are going to take into consideration multiple observations recorded sequentially over time - e.g. determine pollution levels on the atmosphere according to different weather data.

Plainly *based on the literature studied and on the most commonly found references* in recent articles regarding machine learning algorithm implementations for multivariate Time Series Forecasting, we are going to opt for the **implementation of an Artificial Neural Network**. *We have no guarantees that it is the best solution to our problem*, but based on this information the author and the advisor agreed on pursuing a solution based on ANNs due to:

1. their flexibility at assessing patterns;
2. their power as predictive algorithms;
3. their scalability for a larger scope of possible implementations of the behavioral pattern extraction method;
4. the constant flow of new research in the field.

5.1.4 Long Short-Term Memory Networks

As was discussed in section 2.13 of chapter 2, in the field of Artificial Neural Networks, **Recurrent Neural Networks** were created to address Time Series Forecasting problems since regular feed forward neural networks had no notion of time and were not sequenced.

But when implementing large RNNs, authors found that the gradients used to update the weights during backpropagation through time could become either very large or very small - a problem called **exploding or vanishing gradients** which prevented RNNs from learning long-term dependencies.

A largely popular solution was the creation of **Long Short-Term Memory Networks** which could in fact learn long-term dependencies using memory blocks that replaced simple artificial neurons.

Given its well known potential, **we chose to use LSTMs to make our multivariate time series forecasts**.

5.1.5 Summary

After a brief revision of the concepts learned on chapter 2 we have determined that the algorithm which we are going to be implementing as a solution for our time series problems is going to be a **Long Short-Term Memory network**, given:

1. its flexibility at assessing patterns;
2. its power as a predictive algorithm;
3. its ability in Time Series Forecasting;

4. its scalability for a larger scope of possible implementations of the behavioral pattern extraction method;

5. the constant flow of new research in the field;

6. that it is a standard in the industry for Time Series Forecasting.

5.2 Choosing the Dataset

Before implementing the forecasting algorithm we must choose the data the we are going to be using. The data used will determine:

- the **quality** of the predictions;
- the baseline for **how accurate the predictions can be**;
- the **range of the time frame** of the behavioral patterns found;
- the **value of the behavioral correlations found** based on previously scientifically proven cause-effect correlation.

It was important that the dataset contained well studied information so that we could validate the results attained by the network, i.e. we needed to know which behavioral patterns could be detected in the dataset according to previous scientific studies done on the dataset, so that we could validate the behavioral patterns detected by our behavioral pattern extraction method.

5.2.1 University of California, Irvine's Machine Learning Repository

The dataset that we are going to use is going to be a dataset found in the **University of California, Irvine's Machine Learning Repository**. This repository currently maintains 437 data sets as a service to the machine learning community [10]. To this day, new data sets are still being added to the repository and are fairly popular among the machine learning community [3].

The chosen dataset was selected based on two criteria: its *type*, it had to be multivariate time series datasets; and since we had to test the accuracy of behavioral patterns found by our behavioral pattern extraction method *the dataset had to provide data that had already been scientifically studied* so we could validate the extracted behavioral patterns with previously established correlations.

5.2.2 Beijing PM2.5 Data Data Set

The dataset chosen was the Beijing PM2.5 Data Data Set. This dataset contains **hourly data regarding the concentration of PM2.5 in Beijing** from January 1st, 2010 to December 31st, 2014.

PM2.5 are a set of atmospheric particles that have been known to cause respiratory and vascular problems [2]. The data comes from the United States Embassy in Beijing and the Beijing Capital International Airport.

Initially the goal was to use a dataset which contained time series data with records of human behavior, given one of the raised issues was related to the tracking of user activities through tracking applications. Unfortunately we could not find a fitting time series dataset containing logs of human behavior, mainly due to the fact that the datasets found had no scientific information regarding the correlations between the behaviors that allowed us to validate the accuracy of the behavioral patterns extracted with our method. Therefore we opted for a time series dataset that contained other types of scientifically studied behaviors, in this case: the behavior of atmospheric particles and weather conditions.

The attributes of this dataset are the following:

1. **No:** row number
2. **year:** year of the instances
3. **month:** month of the instances
4. **day:** day of the instances
5. **hour:** hour of the instances
6. **pm2.5:** PM2.5 concentration (ug/m^3)
7. **DEWP:** Dew Point, i.e. the atmospheric temperature below which water droplets begin to condense and dew can form. (Degrees Celsius)
8. **TEMP:** the atmospheric temperature (Degrees Celsius)
9. **PRES:** the atmospheric pressure (hPa)
10. **cbwd:** Combined wind direction
11. **Iws:** Cumulated wind speed (m/s)
12. **Is:** Cumulated hours of snow
13. **Ir:** Cumulated hours of rain

The dataset contains missing data which is denoted as *NA* [25].

5.3 Choosing the Technology

Finally, all that is left is to choose is the technology with which we are going to be implementing our solution. Our first focus was:

What are the strongest and most popular programming languages for the development of machine learning algorithms?

The clear winner here is Python.

5.3.1 Python

Python is an interpreted high-level programming language with an emphasis on code readability for clear programming on both small and large scales. It has been **widely adopted by the scientific community** mainly because its ease-of-use enables for quick prototyping and sketching of ideas.

But the major benefit of using Python is that it has some of the **most complete and diverse modules and libraries for machine learning**, such as: pandas, scikit, keras and tensorflow. They are all free and are under the GNU General Public License [33].

Next, in order to create machine learning algorithms, we then had to choose our libraries.

5.3.2 TensorFlow & Keras

One of the most "complete" libraries for the creation of machine learning algorithms such as artificial neural networks is Tensorflow.

Tensorflow was created by the Google Brain Team, a deep learning artificial intelligence research team from Google. It allows for very intricate tweaking and manipulation of virtually every parameter of machine learning algorithms. The biggest drawback when using this library is that it has a relatively steep learning curve and its syntax can sometimes get in the way [28].

Keras is a high-level API for the creation of neural networks. It was developed with a focus on enabling fast experimentation. As their team puts it: *"Being able to go from idea to result with the least possible delay is key to doing good research"*. It can run on top of Tensorflow as an added layer for usability in turn for less control over the network [24].

Given the already dense research background required from machine learning and artificial neural networks, we chose to use Keras to enable faster research and prototyping of solutions.

5.3.3 Other Packages

Along with Keras, we also used the following packages (along with other minor ones):

- **Pandas:** for data manipulation and data analysis
- **matplotlib:** for the plotting of some of the plots used
- **Scikit-learn:** for multiple machine learning operations

Chapter 6

Forecasting Algorithm

Contents

In this chapter we will be creating the forecasting algorithm which will be predicting the concentration levels of PM2.5 in Beijing based on other atmospheric behaviors. We will start by first preparing the data that we have for Time Series Forecasting. Then, we will setup a standard Long Short-Term Memory network and establish a performance baseline.

Finally we will iteratively test and tweak each parameter in the neural network until we achieve a good performance.

6.1 Data Preparation

As we said, before implementing the behavior forecasting algorithm we must first prepare the data. Data preparation is a necessary step for any predictive task. The same dataset can be approached from multiple different angles, according to the solution that is being implemented. Therefore, we need to *polish* the dataset so that it can be used by our forecasting algorithm.

6.1.1 Basic Data Preparation

The first 5 lines of the "Beijing PM2.5 Data Data Set" .csv data set are the following:

```
1 No,year,month,day,hour,pm2.5,DEWP,TEMP,PRES,cbwd,Iws,Is,Ir
2 1,2010,1,1,0,NA,-21,-11,1021,NW,1.79,0,0
3 2,2010,1,1,1,NA,-21,-12,1020,NW,4.92,0,0
4 3,2010,1,1,2,NA,-21,-11,1019,NW,6.71,0,0
5 4,2010,1,1,3,NA,-21,-14,1019,NW,9.84,0,0
```

The first step is to compile the date-time information into a single date-time attribute so that we can use it as an index in the Pandas library. Scanning the file revealed that within the first 24 hours the values for pm2.5 are missing (NA). That means that we will need to remove the first row of data. There are also a few scattered “NA” values later in the dataset; we can mark them with 0 values for now.

In sum the initial data preparation is going to look like this:

1. **Compile** the "year", "month", "day" and "hour" attributes as a **new attribute called "date"**;
2. **Remove the "No" attribute** since the "date" attribute will become the new index;
3. **Replace the "NA"** values with 0;
4. **Remove the first 24 hours** of instances;
5. OPTIONAL: change the names of some of the attributes to make them easier to read

After creating a simple preparation algorithm and running it to apply these changes to the dataset, the first 5 lines of the dataset are:

```
1 date,pollution,dew,temp,press,wnd_dir,wnd_spd,snow,rain
2 2010-01-02 00:00:00,129.0,-16,-4.0,1020.0,SE,1.79,0,0
3 2010-01-02 01:00:00,148.0,-15,-4.0,1020.0,SE,2.68,0,0
4 2010-01-02 02:00:00,159.0,-11,-5.0,1021.0,SE,3.57,0,0
5 2010-01-02 03:00:00,181.0,-7,-5.0,1022.0,SE,5.36,1,0
```

As we have previously referred, our initial goal was to use data of human behavior that resembled the logging of user activity in a tracking application, but we were unable to find a suitable dataset. So, we decided to change the context of the dataset: we framed it as a more short-term dataset in comparison with what was originally intended.

Tracking application user’s are more likely to have a daily tracking behavior, instead of an hourly tracking behavior - as recorded in the PM2.5 dataset. Not only that, but tracking applications, much like any other non-social media application, are not used for such extended periods of time as their social counterparts [21].

So, we decided to **select only the initial 180 hours** for our tests, because it was a better representation of a tracking application user’s tracking its behaviors once a day for 6 months - if we consider that each hour represents one day in recorded tracking application activity.

Finally, we removed some the attributes from the dataset that were either not relevant for the work developed or had no significant value variation throughout the selected time interval.

Therefore we further adjusted the dataset by:

1. **replacing the "date" values** with ordered incremental values as if it was referencing **each hour since the start of the tracking activity**;

2. **Reducing the time interval to 180 hours** making it so that the dataset used will be from January 2nd, 2010 at 00:00 to January 9th, 2010 at 12:00;

3. **removing the "wind direction" attribute** because it was more of a compound behavior that had to be connected with the wind speed instead of being a more standalone behavior - and the behavioral pattern extraction method developed only allows for behavior-to-behavior correlation evaluations, no compound behaviors-to-behavior evaluations;

4. **removing the "rain" attribute** because it showed minute changes throughout the whole dataset and in the first 2500 instances showed no changes at all;

After making these final basic data preparations we have a data set whose first 5 lines are:

```
1 date,pollution,dew,temp,press,wnd_spd,snow
2 1,129,-16,-4,1020,1.79,0
3 2,148,-15,-4,1020,2.68,0
4 3,159,-11,-5,1021,3.57,0
5 4,181,-7,-5,1022,5.36,1
```

It is important to understand that these adjustments may look intrusive but in fact, they will not have too much of an impact in the search for our solution because time is relative to the representation that we give it. **The data values will remain unaltered and scientifically backed correlations will remain constant**, therefore we can still extract and evaluate any behavioral pattern found in the dataset, independent of the time window.

Ideally we would be using a more appropriate dataset that would not need any manipulation, e.g. a time series dataset with the log of human behaviors. But given the circumstances in which our work was developed, this was the most fitting solution.

6.1.2 LSTM Data Preparation

We are now on the final steps of the data preparation, we just need to prepare the pollution dataset for the LSTM. This involves **framing the dataset as a supervised learning problem** and **normalizing the input variables**.

We will frame the supervised learning problem as predicting the pollution at the current hour (t) given the pollution measurement and weather conditions at the prior time steps. The behavioral pattern extraction method developed in the next chapter will be able to detect behavioral correlations between any two behaviors, but in order to know how behaviors are influencing our "target behavior", we must first be able to make good predictions on that target behavior and have **one network for each target behavior**. So for now, we are only going to be predicting the PM2.5 concentration level (target behavior), which we will call "**measurement**", and the other attributes of the dataset will be called "**events**".

Next, all **features are normalized** and the **dataset is transformed into a supervised learning problem**.

Normalizing means that we are going to rescale every feature to a common scale. In this case every feature will fit in a 0 to 1 scale in which 0 represents the minimum value of the feature and 1 represents the maximum value of the feature within the dataset.

The ***dataset transformation*** is done by **removing the "date" attribute** because LSTMs time-series forecasting are not concerned with whether or not the time interval between instances is classified as months, weeks, days or hours; it just assumes that every instance (t) in the dataset is as close in time to the next instance $(t+1)$ as to the previous instance $(t-1)$. This means that there is no need to index the instances since the LSTM assumes a time-series sequence.

Before wrapping the data preparation we just need one more thing. We must split the dataset into **training data** and **testing data**. In standard conventions the training and testing data split is usually 70% and 30% respectively. This split is standard because most forecasting problems revolve around forecasting very small intervals such as 10 to 50 instances after the current instance and so, people choose to use the most amount of data possible to train the models. But since we have plenty of data and the actual applications of this thesis' work can vary tremendously depending on how they will be implemented, we are going to stick with the approach that is most similar to "a user tracking activity through a tracking application" (considering that 1 hour of this dataset can correspond to 1 day of human behavior). For that simple reason we chose to do a **50-50 split and have 90 training hours and 90 testing hours**.

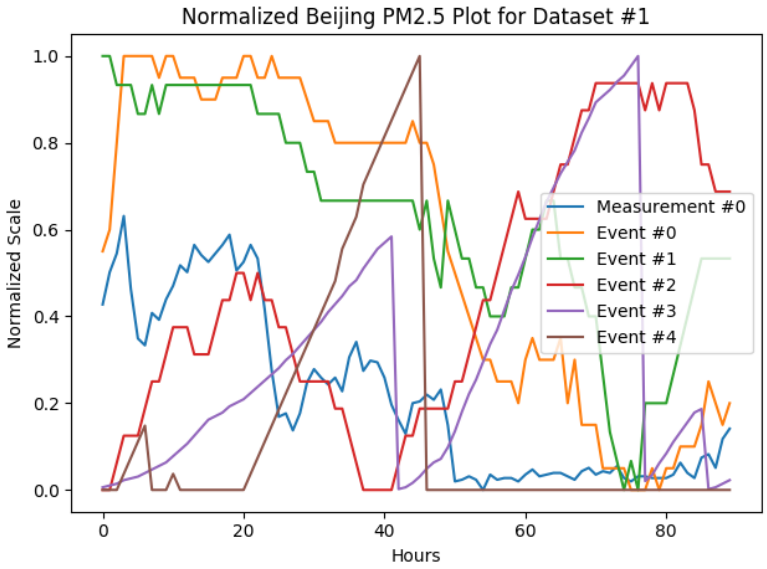

Figure 6.1: Normalized attributes of the first 90 training hours from the Beijing PM2.5 Dataset on a scale of 0 to 1.

In figure 6.1 we can visualize each attribute's variations in a normalized scale, in which 0 represents the minimum value of the attribute and 1 represents the maximum value of the attribute within the dataset:

- **Measurement #0:** pollution, i.e. the PM2.5 concentration (ug/m^3)

- **Event #0:** dew, i.e. the dew point (°C)
- **Event #1:** temp, i.e. the atmospheric temperature (°C)
- **Event #2:** press, i.e. the atmospheric pressure (hPa)
- **Event #3:** wnd_spd, i.e. the cumulated wind speed (m/s)
- **Event #4:** snow, i.e. the cumulated hours of snow

6.2 Implementation & Testing

Now that we have prepared our data, we can start to create an Artificial Neural Network for forecasting the PM2.5 concentration levels in Beijing's atmosphere during the peak if the winter of 2010, based on the previously described attributes. We will start by first setting the topology and training parameters of the network in the most standard way possible to create a baseline. After that, we will tweak each parameter individually and test different hypothesis until we achieve a good performance level.

Given the focus of our work being the general performance of our behavioral pattern extraction method, we will not be optimizing our neural network to a very large extent. We only need to make sure that the predicted values have an average error that meets Requirement 1 from section 4.5. With the pollution level of PM2.5 ranging from around 20 to 280 (as made visible in figure 6.2) we have determined that an RMSE of 39 will be our maximal target.

- **Number of Layers:** 2 - one input layer and one output layer;
- **Number of Neurons/Memory Blocks in each layer:**
  - Input Layer: 6 - one for each attribute of the dataset;
  - Output Layer: 1 - one for each value we want to predict, in this case we are only trying to predict the pollution level;
- **Number of Epochs:** 50, we will start with a low number and then increment it as we go;
- **Batch Size:** 90 - literature says that only when we have limited resources should we reduce the batch size, the larger the amount of samples per batch the better because we are taking into consideration more samples at a time, therefore we are going to be using all instances;

After setting up the initial topology of the network, we ran the neural network and obtained a RMSE of 89.988 as our baseline. In figure 6.2 we can see that the prediction was not even close to the actual values of the testing data.

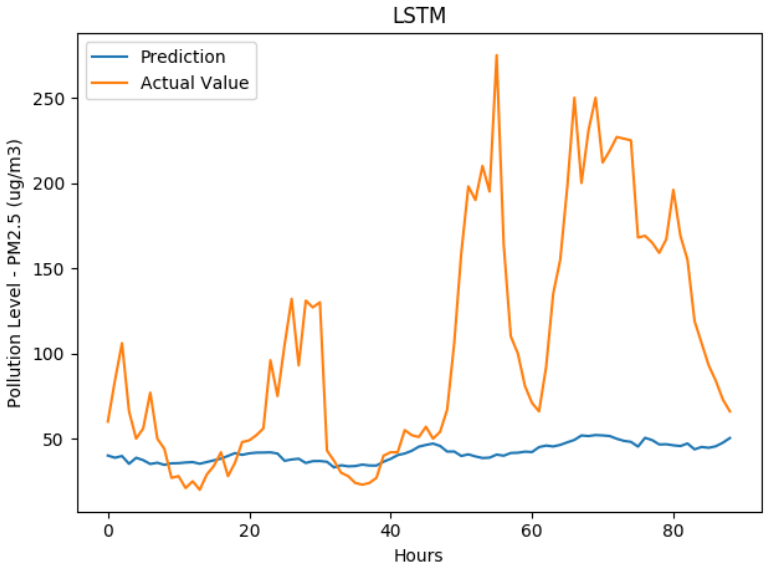

Figure 6.2: Baseline graph for the predictive performance of the forecasting algorithm, when forecasting 90 hours after the final training time.

6.2.1 Epochs

The first thing that we should adjust are the features that most influence how much the neural network is actually learning. Arguably, the most relevant learning factor is how many times the network actually reads the data before coming to a conclusion - the number of **epochs**.

Table 6.1: LSTM performance based on epochs.

Epochs	RMSE	Elapsed Time (sec)
50	89.988	1
100	81.993	1
200	67.815	2
400	45.535	2
800	38.514	4
1600	35.421	7
3200	**31.711**	**13**
6400	34.22	26
12800	33.646	51
25600	29.315	103
51200	31.98	200

We created a set of tests in which we ran the neural network with an incremented amount of epochs and then recorded the RMSE and the amount of seconds it took for the network to predict new values. **Each epoch increment was a two fold increment and we stopped when we did not see any relevant increase in performance with an exponential increase of epochs**.

As we can see from the value highlighted in bold on table 6.1, 3200 epochs is the 2nd best performing network with a difference of just 2.396 in RMSE. But this lack of performance is compensated by being nearly 8 times faster than the best performing network (25600 epochs), making **3200 our choice for the number of epochs that our LSTM will have**.

6.2.2 Neurons

For now, we only have two layers. One of which, the output layer, cannot be changed since there only needs to be 1 neuron per output. So we can only change the input layer.

Since in standard conventions we should have at least one neuron per attribute of the dataset we can only increase the amount of neurons. We will use the same two fold incremental method used for the adjustment of the number of epochs, but will start by adding just 1 neuron and stop when we see no relevant increase in performance with an exponential increase in the number of neurons. We will also add the RMSE and Elapsed Time values of our network with 3200 epochs at the top of the table for better value referencing.

Table 6.2: LSTM performance based on number of added neurons to the input layer.

Added Neurons	RMSE	Elapsed Time (sec)
0	31.711	13
1	**30.202**	**13**
2	30.402	13
4	31.224	14
8	31.924	13
16	31.848	14
32	36.778	14
64	36.254	15
128	35.326	21

As we can see from the value highlighted in bold on table 6.2, there was no increase of performance for more than 1 added neuron. There was a slight increase in performance with no visible decrease in speed when a single neuron was added. With these results **we will be adding 1 neuron to the input layer**.

6.2.3 Layers

Recurrent Neural Networks, such as the Long Short-Term Memory network being used for our forecasting algorithm, add recurrent behavior to the neurons as well as the ability for them to communicate with other neurons in the "same layer" as seen on section 2.13. This means that we don't necessarily have a *layered* architecture with a restricted amount of neurons running in parallel. When unfolding a recurrent neural network with a "single layer of 6 neurons", particularly an LSTM, we will have multiple communications between multiple neurons and not necessarily 6 neurons feeding the output layer directly right on the first loop.

This means that adding layers to network should not make much of a difference, similarly to what happened when we added neurons to the input layer. Still, we will try to add more LSTM layers to our network to see if we can increase performance. Each new layers will have the same amount of neurons as the input layer and we will add the RMSE and Elapsed Time values of our network with 3200 epochs and 1 added neuron to the input layer at the top of the table for better value referencing. We will use the same two fold method that we used in the previous adjustments and then discuss the results.

Table 6.3: LSTM performance based on number of added 6 neuron layers.

Added Layers	RMSE	Elapsed Time (sec)
0	**30.202**	**13**
1	30.827	20
2	34.281	28
4	30.386	42
8	32.371	73
16	80.324	133

As was expected the recurrent nature of the network made the increase in amount of layers apparently irrelevant. **The best performing network was the network that had no added layers**.

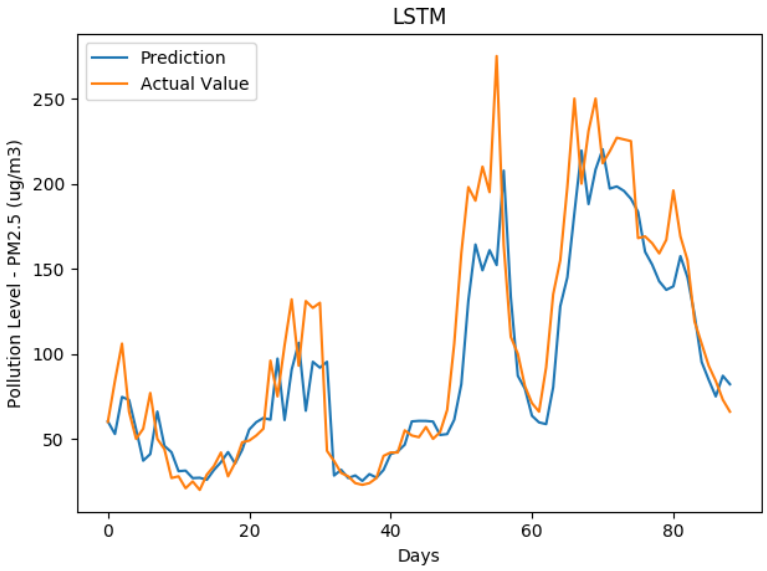

Figure 6.3: Graph of the predictive performance obtained by the optimized LSTM in forecasting 90 hours after the final training time.

6.3 Conclusion

By using standard optimization protocols, we have managed to create a Long Short-Term Memory Network which predicts PM2.5 levels with relative accuracy. As we can see from figure 6.3,

the predicted function and the actual underlying function of the data very similar in shape, despite being slightly delayed by 1-2 days. The largest errors come from predictions that underestimated very large increases in value as seen from hours 20 to 30, 50 to 60, and 60 to 70.

The network is composed of 2 layers: a recurrent input layer with an extra neuron, and a single output layer. When training the network, all of the 90 instances will be batched and looked at the same time 3200 times before the network makes a prediction.

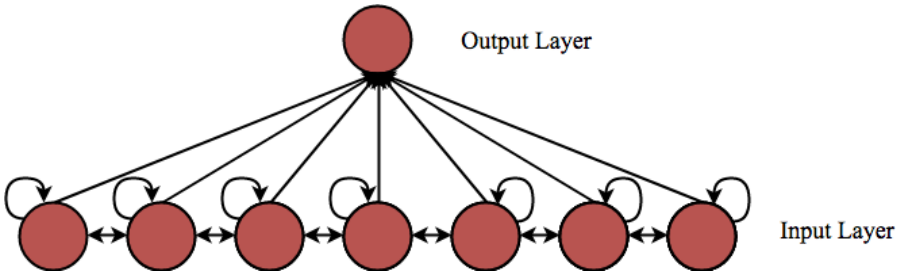

Figure 6.4: A model of the LSTM network created for the forecasting algorithm, containing a long short-term memory input layer with 7 neurons and a single output layer.

Evidently, a better performance would mean a better accuracy in detecting behavioral patterns - and meticulous optimization is advised in future work - but for the detection of simple correlations in average value variations between two behaviors, the general similarities in shape of the predicted function are sufficient for a proof of concept.

Overall, **the network seems to be making sufficiently good predictions** (RMSE under 39), knowing that no meticulous optimization method was used other than standard methods used through testing and tweaking of the major components of the network, as suggested on the answer to Research Question 1, from section 4.5.

Chapter 7

Behavioral Pattern Extraction Methods

Contents

In this chapter we will be creating two simple methods that will enable us to recognize rudimentary behavioral patterns from data. The tests for evaluating these methods will be ran on the Long Short-Term Memory Network created in the previous chapter. We will be extracting the behavioral patterns detected by the network and then evaluate them according to how similar they are to the underlying scientific laws which regulate the natural behavior recorded by the data.

7.1 Overview

There are a multitude of different characteristics that can be used to define a behavioral pattern, for example:

- **One-To-One Relation:** whenever A is this, B is that;
- **Many-to-One Relation:** whenever A and B are this, C is that;
- **Many-To-Many Relation:** whenever A and B are this, C and D are that;
- **Seasonal Occurrence:** every X days, A is that;
- **Immediate Response:** right when A is this, B is that;
- **Delayed Response:** whenever A is this, X time later B is that;

The methods developed will only focus on finding general **one-to-one behavioral patterns that can be found in a single time frame**. These patterns do not take into consideration any

behavioral: seasonality; response time or response intensity. Our goal was to create a rudimentary way of approaching behavioral pattern extraction.

As we will be able to better understand throughout the development of this chapter, and as was previously referred throughout this thesis: this field concerned with the extraction of behavioral patterns from artificial neural networks, despite not being new, still has not gained too much traction. There are few papers on the subject, and the ones that do publicly exist do not go very deep into it.

Because of the scope and the resources available for our work, we decided to only scratch a bit of its surface. In order to begin any kind of new work, the greatest overhead imposed by this field is the amount of research that has to be done. The field of machine learning, and its subfield of artificial neural networks are very dense, as should be expected from any specific scientific field at this level.

Having said this, in the following sections, we will be continuing the implementation of our final solution. During this part we will discuss any decisions and assumptions made when searching for the best possible solution.

As was previously stated on chapter 4 our methods will detect two general behavioral patterns:

1. **Behavior Connection Weight Pattern:** how much the changes in a behavior induce change in another behavior, i.e. if a change in behavior A is related with a small or a big change in behavior B;

2. **Average Linear Correlation Direction Pattern:** how the variation in a behavior is correlated with the variations of another behavior, i.e. if an increase or decrease in behavior A is related with an increase or decrease in behavior B;

We will start by describing the Behavior Connection Weight pattern.

7.1.1 Behavior Connection Weight Pattern

The approach behind the Behavior Connection Weight pattern draws resemblance with previously documented approaches in the field. The goal is to **detect how much each variable from the dataset (also known as events) is contributing to the prediction of a target behavior (also known as the measurement)**. In other words, we are trying to find the weight of each input variable in the prediction of the output variable.

In the State-Of-The-Art chapter (chapter 3) we have talked about many different ways of approaching this problem namely: the connection weight method created by Olden et al. and different sensitivity analysis methods.

According to the literature, the connection weight method was the best at correctly estimating the weight of each variable with a 92% accuracy, while the other sensitivity methods showed around 70% accuracy.

The connection weight approach created by Olden et al. was developed in a simple feed forward Artificial Neural Network. The network would predict a certain value and then the connection weight method would evaluate each variable weight by extracting a series of connection weights spread across the network (as made clear in subsection 3.2.1).

This process can be easily transposed into another similar feed forward artificial neural network; but not so much into a recurrent long short-term memory network. Long Short-Term Memory networks have a much more complex architecture than a regular artificial neural network. Mainly due to two aspects:

1. **LSTMs are recurrent:** as was previously stated in chapter 2, recurrent neural networks' neurons do not simply communicate with the neurons from the following layer - they can communicate with neurons from the same layer, the previous layer, and even to themselves - this makes it much more difficult to extract the relevant connection weights;

2. **LSTMs do not have regular artificial neurons:** in order to solve the exploding and vanishing gradient problem found in regular recurrent neural networks, LSTMs have an "upgraded" version of the regular neuron - these new neurons are made up of memory blocks and a gate system which controls the flow of information - this adds an extra layer of complexity to the already complex extraction of connection weights from RNNs.

For this reason, using the connection weight method for the extraction of the behavior connection weight pattern would be unfeasible given the amount of extra time necessary to create an adapted implementation of the method for LSTMs.

Therefore we decided to go for the second best option and **implement a sensitivity analysis** method which conveys 70% accuracy to our Behavior Connection Weight Pattern extraction method.

7.1.2 Average Linear Correlation Direction Pattern

The goal of the Average Linear Correlation Direction pattern is to **identify the *direction* of the connection between an event and a measurement**. In other words, the method attempts to understand if the increase or decrease of an attribute's value causes an increase or decrease in another attribute's value, i.e. if there is a proportional or inverse linear correlation.

The method itself will be looking for the average direction of the correlation, since it will try to understand what is the average linear trend of the correlation between these two behaviors. For example, when asserting that there is a proportional linear correlation between behavior A and behavior B, the method is saying that on average, during the majority of the time interval, when A increases, B increases.

But the method also does another thing: in case the neural network **recognizes both proportional and inverse correlations in that time frame, then the method will assert that the correlation may not be linear**. Because in some cases, independently of A increasing or decreasing, B may increase or decrease in both variations of A. This does not necessarily mean that A's

behavior is not correlated with B - e.g. it can simply mean that whenever A changes X number of units, B increases or decreases. In this example, the correlation does exist, but it is not linear.

7.2 Implementation

Now that we have talked about the goal of each pattern, we will proceed with the implementation of the methods that will enable their extraction.

7.2.1 Behavior Connection Weight Pattern

The foundation for the method that will extract this pattern is to understand how much the network is relying on each attribute to make correct predictions. The way that we can do this is by using a *brute force* sensitivity analysis method. Using the same neural network we do the following:

1. run the original model on the original dataset and determine its RMSE (the average distance error between the predicted and the actual values);
2. completely remove from the dataset the attribute (event) that we want to evaluate;
3. create a new model using the new dataset and determine its RMSE.

After this process is completed we can then determine the weight of the connection and how much the network is relying on the removed attribute to make its predictions, by subtracting the original RMSE from the RMSE of the new model.

1. the method **repeats this process for all the events in the database and collects their RMSE differences** - which determine the weight of their connections to the measurement;
2. each event is then ranked according to where it stays in comparison with all of the other events in the dataset.

This ranking method contributes to an easier comprehension of the influence that each behavior may have on the target behavior. This correlation, does not necessarily mean causation, but it serves as a possible indication of such a relationship.

7.2.2 Average Linear Correlation Direction Pattern

As was stated before, the goal of this extraction method is to determine if there is a linear correlation between two behaviors; and in case there is, determine if it is a proportional or inverse linear correlation.

The implementation of this method is based on the Traditional Sensitivity Analysis method: **varying a behavior throughout its whole range and understanding how that affects another behavior**. In order to assess if there is a linear correlation between a behavior (event) and the predicted behavior (measurement) our algorithm does the following:

1. run the original model on the original testing portion of the dataset and determine its mean (*ogMean*);

2. maximize all of the values of the event in the test portion of the dataset;

3. run the original model on the new dataset and determine the new prediction's mean (*maxEventMean*);

What we are doing here is, we are telling our neural network to learn the patterns from the original training data. Then we maximize the values of the event that we want to evaluate and tell the network to make predictions based on these maximized values.

We do this because the network will apply what it has learned about the event and the measurement's relationship. **If it has learned that there is a clear linear correlation between them, then it will be visible on its predictions**. Therefore, after the prediction, we study the network's behavior by comparing how it predicted with the new dataset. This comparison is done in an expression similar to the following pseudo-code:

```
if (maxEventMean > ogMean)
  then "there may be a proportional linear correlation"

else if (maxEventMean < ogMean)
  then "there may be an inverse linear correlation"
```

It basically means that: if the predicted measurement value **generally increases** when the values of the attribute are maxed out, then the network could be **detecting a proportional linear correlation** between the event and the measurement; on the other hand, if the predicted measurement value **generally decreases** when the values of the attribute are maxed out, then the network could be **detecting an inverse linear correlation**. After this comparison the algorithm:

4. minimizes all of the values of the event in the test portion of the dataset;

5. runs the original model on the new dataset and determines the new prediction's mean (*minEventMean*)

6. compare all three means: *ogMean*, *maxEventMean* and *minEventMean*.

This part is meant to either confirm and determine the type of linear correlation, or refute that there is any linear correlation being detected by the algorithm. This comparison is done using an expression similar to the following pseudo-code:

```
if (maxEventMean > ogMean AND minEventMean < ogMean)
  then "there is a PROPORTIONAL linear correlation"

if (maxEventMean < ogMean AND minEventMean > ogMean)
  then "there is an INVERSE linear correlation"
```

```
6
7  if (maxEventMean > ogMean AND minEventMean > ogMean)
8    then "there is NO linear correlation"
9
10 if (maxEventMean < ogMean AND minEventMean < ogMean)
11   then "there is NO linear correlation"
```

What this does is, it analysis the network's behavior to **check if the initial guesses continue to make sense** when the values of the event are minimized. If for any reason the analysis tells us that the network is saying that the maximization or minimization of the event has the same effect on the measurement, then we can affirm that the network is saying that there is no clear linear correlation between the two behaviors.

7.3 Testing

Now that we have implemented both methods for the behavioral pattern extraction, we can test each method and assess its performance. In order to do so, we must first **create the foundation for a comparison method** which will allow us to understand how accurate each method is.

For that we will be using a series of studies which used statistical methods to understand how weather conditions have been affecting the concentration of PM2.5 in Beijing from 2008 to 2017. The first two studies by Huang et al. and Yin et al. are dedicated to studying the relationship between meteorological conditions and the concentration of PM2.5 in Beijing [14, 40]. The last two studies by Chen et al. and Yang et al. study this relationship throughout China and talk about specific regions and metropolitan cities such as Beijing [8, 41].

All of the studies divide each year into four seasons: spring from March to May; summer from June to August; autumn from September to November; and winter from December to February. This will mean that, since the portion of the dataset used spans from January 2nd, 2010 at 00:00 to January 9th, 2010 at 12:00, most of the evaluations will be focusing on peak winter season.

Furthermore, there is one attribute that has a different interpretation in these studies: *dew point*. Dew point refers to the temperature below which water droplets begin to condense, i.e. it is the condensation point from which water in its gas state starts to turn into liquid state. Every single one of the studies, replaces the dew point attribute with **relative humidity**. This is because **dew point temperature is a measure of humidity**. Humidity is a percentage value that can be obtained with temperature and the dew point:

> *The closer the dew point is to the current temperature, the higher the relative humidity. A relative humidity of 100% indicates the dew point is equal to the current temperature and that the air is maximally saturated with water[18].*

Therefore, the higher the dew point, the higher the humidity in a seasonal analysis where temperature is stable throughout each season - particularly in peak months of the season such as

the one being used in our dataset. For those reasons and for comprehension purposes, the following parts of this section will **refer to *dew point* as *humidity***.

7.3.1 Behavior Connection Weight Pattern

In order to test the accuracy of the connection weights extracted by the behavioral connection weight extraction method, the algorithm will do the following:

1. each attribute in the dataset will be evaluated and its connection weight with the PM2.5 concentration level will be extracted;
2. all attribute connection weights will be collected and sorted in descending order of weight value;
3. the order of the attributes will then be discussed in comparison with the known scientifically backed order.

7.3.1.1 Scientific Consensus

According to the literature, the general consensus for the relevance of each meteorological variable is that, wind and humidity show the strongest correlation levels with PM2.5 levels, temperature being closely behind along with pressure and snow or other types of precipitation being the least relevant of the group.

On average, wind and humidity are very close to each other, having both an average correlation coefficient of 0.58 in magnitude all year round. But as was detected by Chen et al. in heavily polluted winters, like the winter of 2010, **the most dominant meteorological driver for PM2.5 concentration is the wind** throughout the whole North of China, including Beijing [8]. That helps us identify wind as being the most relevant meteorological condition of all.

Between temperature and pressure, the study made by Yin et al. reaches the conclusion that **pressure is more relevant then temperature** with an average correlation coefficient of 0.47 and 0.40 respectively [40]. And finally, snow and other precipitation are the least relevant with a correlation coefficient of around 0.11. That closes the correlation rank between all of the attributes.

It is important to refer that these coefficients are not taking *direction* into consideration. The current values only represent the connection weight, between each of the attributes and the PM2.5 concentration level. Direction will be used to test the *average linear correlation direction pattern*.

7.3.1.2 Discussion

By running the neural network and using the implemented method, the connection weights are extracted and ranked. On table 7.1 we can observe the real ranks of each event, the event ranks extracted with the method as well as the RMSE differences used by the method to determine the connection weights between the events and the measurement.

Table 7.1: Correlation rank of the relevance of each meteorological behavior when forecasting PM2.5 concentration levels according to the results extracted by the Behavior Connection Weight method, with the RMSE difference used to weigh each connection.

Rank	Actual Event	Extracted Event	RMSE Difference
1	Wind	Wind	2.4328
2	Humidity	Humidity	1.7667
3	Pressure	Pressure	0.2040
4	Temperature	Snow	-0.5267
5	Snow	Temperature	-2.7929

Despite the time window of the dataset being relatively small and the training data only encompassing a total of 80 hours out of the 722 existent in a winter season, **the method showed good results by determining the correct rank for 3 out of the 5 attributes**. This coincides with the results obtained by Olden et al. regarding the accuracy of sensitivity analysis methods for connection weight ranking - in which the Behavior Connection Weight pattern extraction method is based on. The extraction method showed an accuracy of 60% which comes only slightly short of the expected 70%. This result is comprehensible since the next best result would have been to switch Snow for Temperature and then the method would have had 100% accuracy for this dataset.

The only problem found is that in the actual event rankings, the temperature attribute had a far heavier connection with the PM2.5 concentration than snow (0.40 and 0.11). While **on the extracted ranking the temperature was far below the ranking of the snow attribute**. The RMSE difference between the original error and the error obtained after each attribute was individually extracted from the dataset, shows that the temperature had a far lower weight than snow, with an RMSE of -2.7929 for temperature and -0.5267 for snow. A possible reason for this could be that the neural network was not fully comprehending the relationship between each attribute and the PM2.5 concentration level.

A negative RMSE difference value tells us that, when removed from the dataset, these attributes actually decreased the original RMSE. This means that **these attributes where confusing the network** and when removed, helped the neural network make more sense of the data. There are a lot of reasons for why this could be happening, the most likely one being that there wasn't sufficient amounts of optimization: the closer the original RMSE is to 0.0, the smaller are the chances for a dataset *perturbation* to reduce confusion of the network. This problem could be fixed by utilizing more complex and focused optimization methods, but given the scope of the thesis being the creation of pattern extraction methods without super-optimization for the datasets used: having the method met the expected outcome from the underlying sensitivity analysis methods used, **the results are deemed satisfactory for the conclusion of the implementation of this method**.

7.3.2 Average Linear Correlation Direction Pattern

For the evaluation of the extraction of the Average Linear Correlation Direction Pattern, we will be using the extraction method to understand if there is any linear correlation between each event and the pollution level, and its direction - in case there is a clear linear correlation. Then we will compare the results to the ones obtained by the literature of the field and reach our conclusions.

7.3.2.1 Scientific Consensus

There is a general consensus about two specific natural laws that control the pollution levels in Beijing given its climate and geographical location. The biggest meteorological contributer to a considerable decrement in the pm2.5 concentration levels is the wind. **The stronger the wind, the more dispersion of particles, the smaller the concentration of pollutants**.

Despite pollution levels having a general proportional correlation with the increase in temperature in most places in the world, given the geographical location of Beijing, increases in temperature usually increase the airflow between the lowest and the highest levels of the atmosphere. This causes increases in wind speed, making **increases in temperature correlated with decreases in pm2.5 concentration levels**.

Most of the studies referenced in this thesis agree with those rules. They also agree that **increases in snow and other precipitation decrease the amount of pm2.5**, giving them a clear inverse linear correlation.

The most controversial meteorological condition is humidity. Despite some more accurate recent studies made about the effect of meteorological conditions in pm2.5 concentration in Beijing from 2015 onwards stating that humidity has a proportional correlation with pm2.5 concentration, they have also acknowledged that this positive correlation showed that higher humidity levels often came associated with windless, cloudy, and insufficiently sunny days, which aggravated the accumulation and chemical reaction of pollutants [40]. But in contrast, studies made on heavily polluted winters from 2008 to 2012 (including the one in our dataset) showed an inverse correlation. Given that the newer studies did not take into consideration the winter of 2010, and given that most cases in which humidity correlation was positive were linked with other particular weather conditions: we have decided that we will consider **humidity to have an inverse linear correlation with the concentration of pm2.5** as the most fitting rule.

Pressure was the one that did not show a particularly accentuated linear correlation with the concentration of pm2.5 particles [40]. A linear function could be inferred, but, the actual underlying function extracted from data was strongly linear.

7.3.2.2 Discussion

After extracting the average linear correlation from our data (table 7.2) we once again get the same accuracy level as the previous method: the initial 3 correlations were correct and the following 2 were not, resulting in a 60% accuracy.

Table 7.2: Comparison between the existence of a clear linear correlation between meteorological conditions and the concentration level of pm2.5 according to literature and according the method developed.

Condition	Actual Linear Correlation	Extracted Linear Correlation
Wind	Inverse	Inverse
Humidity	Inverse	Inverse
Pressure	No	No
Temperature	Inverse	Proportional
Snow	Inverse	No

The lack of precision for the prediction of the correlation of the temperature and snow attributes can also be correlated with the negative RMSE difference shown in table 7.1, where **the network seems to not be able to make absolute sense of all the attributes in the data**.

This once again points to the possible culprit being the optimization methods used. The sensitivity analysis methods used could also be further improved upon, if we were to create a method similar to the one suggested by Olden et al. [23]. Still given the scope of the thesis, having the method met the expected outcome from the underlying sensitivity analysis methods used, **the results are deemed satisfactory** for the conclusion of the implementation of this method.

7.4 Summary

In general, the behavioral pattern extraction methods developed showed good results. They were both able to correctly extract patterns for 3 out of the 5 meteorological conditions, which coincides with the average performance levels of the sensitivity analysis methods from which they were based on. Coincidently, all correct and incorrect predictions were made on the same events. This reinforces the idea that the most likely cause for incorrect predictions might have been the suboptimal optimization methods used. Another reason could be that the dataset portion used for the tests did not have sufficient amounts of data for the detection of these patterns.

Overall, the behavioral pattern extraction methods were successful and the results were satisfactory.

Chapter 8

Conclusions

Contents

This thesis started with the presentation of two distinct problems. The first problem arose from the observation that Time Series Forecasting has been seeing tremendous evolution thanks to Artificial Intelligence and Machine Learning. But these advancements have not equally translated into the development of the Time Series Analysis field, which continues to use relatively traditional mathematical methods for behavioral pattern recognition.

The second problem was that tracking applications which attempt to help users achieve their goals, are not taking into consideration each user's individually specific needs when providing tips and advices. Instead they are labeling them according to a "model subject" with whom they bear most resemblance to, and then providing them with tips and advices accordingly. This approach can lead to possible harm to users given neglected factors that are not being taken into consideration when labeling the user.

As we progressed with the analysis of both of these problems, their possible solutions seemed to be converging into a similar direction. The development of a more modern method which fused new machine learning technologies with Time Series Analysis could become the solution for finding an alternative implementation for the strongly biased paradigm that could possibly harm tracking application users.

And so, a solution proposal was formulated. The idea was to create behavioral pattern extraction methods that would extract the behavioral patterns recognized by machine learning algorithms and present them in an explicit format that could easily be understood. This way, the method itself would be the embodiment of a more modern method which fused new machine learning technologies with Time Series Analysis, and its implementation in a tracking application would allow

for the user to understand how individual behaviors were influencing their progress towards their goals.

8.1 Results

During the research and development phase, it was possible to understand that the density of the fields which were encompassed by the academic scope of the work would prove to take a substantial portion of the estimated development time. Therefore, the solution created had to resort to the implementation of methods which were not optimal, but were manageable and provided enough substance for the reaching of strong and valid conclusions.

Initially, we created a long short-term memory network which met the initial Requirement 1, by having a predictive performance whose root-mean-square error was around 11.5% of the range of the predicted attribute's values in the training data. The network was created with standard optimization methods which tuned basic training and topological aspects.

Then, two methods were created. The first one, allowed for the understanding of how much each behavior was influencing the prediction of a target behavior and how these behaviors ranked against each other. The second one, allowed for the understanding of how each behavior was linearly correlated with a target behavior. If a linear correlation was predicted to exist, then the method would tell if the relationship between those two behaviors was proportional or inverse.

Both methods showed good results, having a 60% accuracy at extracting correct behavioral patterns. This means that the following two requirements were met given that the methods were successful at extracting behavioral patterns, and the percentage of accurately predicted patterns matched the average expected percentage of accurately predicted patterns by the methods from which they were based off.

Finally, the method developed did not make any assumptions on the data. The methods themselves relied simply on artificial neural networks and simple statistical methods without ever manipulating or altering the values of the data used. This means that in the end, all requirements were met and the results were deemed satisfactory.

8.2 Contributions

The majority of the work developed was for research and academic purposes and so, the implementations made were created with the goal of allowing for quick prototyping and restructuring of any progress made. Despite it not being fully implemented in an industrial or commercial scenario, the development done was important for the conceptualization of a method that allows for the better understanding of artificial neural networks and how this comprehension can benefit many other fields.

The methods developed allowed us to understand that it is possible to create behavioral pattern extraction methods that can reach similar conclusions to the ones reached by traditional and rigorous statistical methods, by instead extracting behavioral patterns from predictive artificial neural networks.

These methods also proved that there is the possibility to extract accurate behavioral patterns from small time intervals, such as the ones found in tracking applications which record user actions. This opens a future possibility for allowing users to get personalized user experiences specifically adjusted to the users individual needs.

8.3 Future Work

During the work developed, there were some sub-optimal decisions that had to be done given the scope of the work and the resources available. In future work these decisions should be reevaluated and further testing environments should be created.

The first step would be to use datasets with human behavioral patterns recorded, so that we could validate our implementations in the environment for which they were envisioned. Then, we could create more testing environments from these datasets - so that we could assert performance levels with a higher level of confidence. This would require constant optimization for each new dataset or portion of the dataset used, so the creation of an automated neural network optimization method would also be of relevance. Also, the optimization methods should be enhanced as well, to provide more accurate results. There are a multitude of more complex optimization methods that can greatly reduce predictive errors, particularly using lower-level methods for more minute neural network manipulations.

Finally, the methods used for the extraction of the behavioral patterns should be reevaluated. The sensitivity analysis method used for the Behavior Connection Weight pattern extraction could be replaced with an adapted long short-term memory network version of the connection weight method created by Olden et al. - which has shown to have a much higher precision in comparison with the sensitivity analysis method used (92% accuracy in comparison with the sensitivity analysis method's 70% accuracy).

After these optimizations of the work developed, the clear next step would be to create more methods for the extraction of more complex behavioral patterns. For example: extracting many-to-one behavioral patterns with immediate or delayed response times and even taking into consideration response intensity.

8.4 Epilogue

Overall, the development of this thesis served as a tremendous research work which enabled for the comprehension of multiple fields and concepts within the field of Artificial Intelligence. The analysis of multiple machine learning algorithms, artificial neural networks, analysis methods and their respective architectures and components, provided excellent insight on how to approach

future work and research on these fields. In particular, it allowed for a better understanding of which paths to follow in order to continue research on the creation of behavioral pattern extraction methods, namely in long short-term memory networks and other recurrent or feed forward neural networks.

References

[1] Univariate time series models. *e-Handbook of Statistical Methods*, January 2004.

[2] United States Environmental Protection Agency. What is pm2.5 and why you should care | bliss air. Bliss Air. Available at https://blissair.com/what-is-pm-2-5.htm, 2018.

[3] Jason Brownlee. *Deep Learning With Python: Develop Deep Learning Models on Theano and TensorFlow Using Keras*. Machine Learning Mastery, v1.13 edition, 2017.

[4] Jason Brownlee. *Master Machine Learning Algorithms: Discover How They Work and Implement Them From Scratch*. Machine Learning Mastery, v1.13 edition, 2017.

[5] Jason Brownlee. What is time series forecasting? Machine Learning Mastery. Available at https://machinelearningmastery.com/time-series-forecasting/, 2018.

[6] Science Buddies. Steps of the scientific method. Science Buddies. Available at https://www.sciencebuddies.org/science-fair-projects/science-fair/steps-of-the-scientific-method, 2018.

[7] Nikhil Buduma. A deep dive into recurrent neural nets. Nikhil Buduma. Available at http://nikhilbuduma.com/2015/01/11/a-deep-dive-into-recurrent-neural-networks/, 2015.

[8] Ziyue Chen; Xiaoming Xie; Jun Cai; Danlu Chen; Bingbo Gao; Bin He; Nianliang Cheng; and Bing Xu. Understanding meteorological influences on pm2.5 concentrations across china: a temporal and spatial perspective. 2018.

[9] Julian D. Olden; Michael K. Joy; Russel G. Death. An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data. *Ecological Modelling. 178, 389-397*, 2004.

[10] Efi Dheeru, Dua; Karra Taniskidou. UCI machine learning repository, 2017.

[11] Tuples Edu. What is an artificial neuron? Becoming Human. Available at https://becominghuman.ai/what-is-an-artificial-neuron-8b2e421ce42e, 2017.

[12] I.; Lek S. Gevrey, M.; Dimopoulos. Review and comparison of methods to study the contribution of variables in artificial neural netowrk models. *Ecological Modelling. 160, 249-264*, 2003.

[13] Google. Ibm common stock, nyse: Ibm. Google. Available at https://www.google.com/search?q=ibm+stock&ie=utf-8&oe=utf-8&client=firefox-b-ab, 2018.

[14] Fangfang Huang; Xia Li; Chao Wang; Qin Xu; Wei Wang; Yanxia Luo; Lixin Tao; Qi Gao; Jin Guo; Sipeng Chen; Kai Cao; Long Liu; Ni Gao; Xiangtong Liu; Kun Yang; Aoshuang Yan; Xiuhua Guo. Pm2.5 spatiotemporal variations and the relationship with meteorological factors during 2013-2014 in beijing, china. 2015.

[15] MyFitnessPal Helpdesk. How does myfitnesspal calculate my initial goals? MyFitnessPal. Available at http://myfitnesspal.desk.com/customer/en/portal/articles/410332-how-does-myfitnesspal-calculate-my-initial-goals-, 2015.

[16] MyFitnessPal Helpdesk. How does myfitnesspal calculate my initial goals? MyFitnessPal. Available at https://myfitnesspal.desk.com/customer/portal/articles/410332-how-does-myfitnesspal-calculate-my-initial-goals-, 2015.

[17] MyFitnessPal Helpdesk. A message about myfitnesspal's updated nutrition goals. MyFitnessPal. Available at http://myfitnesspal.desk.com/customer/portal/articles/1375583-a-message-about-myfitnesspal-s-updated-nutrition-goals, 2015.

[18] Steve Horstmeyer. Relative humidity....relative to what? the dew point temperature...a better approach. Available at http://www.shorstmeyer.com/wxfaqs/humidity/humidity.html, 2008.

[19] Muhammad Imdadullah. Time series analysis and forecasting. *Time Series Analysis and Forecasting*, January 2004.

[20] Noom Incorporated. Noom: Stop dieting. get life-long results. Noom. Available at https://ww1.noom.com/programs/health-weight/exmain01/?utm_content=exmain01&utm_source=homepage, 2016.

[21] SAG ipl. The 20 most used and downloaded apps in the world 2018. Available at https://blog.sagipl.com/most-used-apps/, 2018.

[22] Deep Learning 4 J. A beginner's guide to recurrent networks and lstms. Deep Learning 4 J. Available at https://deeplearning4j.org/lstm.html, 2017.

[23] Julian D. Olden; Donald A. Jackson. Illuminating the black box: a randomization approach for understanding variable contributions in artificial neural networks. *Ecological Modelling. 154, 135-150*, 2002.

[24] Keras. Keras documentation. Keras. Available at https://keras.io/, 2018.

[25] T.; Guo B.; Li S.; Zhang H.; Zhang S.; Huang H. Liang, X.; Zou and S. X. Chen. Assessing beijing's pm2.5 pollution: severity, weather impact, apec and winter heating. *Proceedings of the Royal Society A, 471, 20150257*, 2015.

[26] Mark P. Mattson. Superior pattern processing is the essence of the evolved human brain. August 2014.

[27] Warren; Walter Pitts McCulloch. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics.*, 1943.

[28] Aakash Nain. Tensorflow or keras? which one should i learn? Medium. Available at https://medium.com/implodinggradients/tensorflow-or-keras-which-one-should-i-learn-5dd7fa3f9ca0, 2017.

[29] Robert Nau. Statistical forecasting: notes on regression and time series analysis. Duke University, Fuqua School of Business. Available at https://people.duke.edu/~rnau/411home.htm, 2015.

[30] NIST/SEMATECH. *e-Handbook of Statistical Methods*. http://www.itl.nist.gov/div898/handbook/, 2013.

[31] Australian Government's Department of Health's National Eating Disorders Collaboration. Eating disorder - understanding the warning signs. Australian Government's Department of Health's National Eating Disorders Collaboration. Available at http://www.nedc.com.au/recognise-the-warning-signs, 2015.

[32] Christopher Olah. Understanding lstm networks. Colah's Blog, Fuqua School of Business. Available at http://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015.

[33] Prince Patel. Why python is the most popular language used for machine learning. Medium. Available at https://medium.com/@UdacityINDIA/why-use-python-for-machine-learning-e4b0b4457a77, 2018.

[34] SAS. Machine learning: what it is and why it matters | sas. SAS. Available at https://www.sas.com/it_it/insights/analytics/machine-learning.html, 2018.

[35] Anna Schaefer. The best calorie counter apps of 2016. Medical News Today. Available at https://www.medicalnewstoday.com/articles/318610.php, 2017.

[36] Sepp Hochreiter; Jürgen Schmidhuber. Long short-term memory. *Neural Computation 9*, 1997.

[37] Amit Shekhar. Understanding the recurrent neural network. Let's Learn AI. Available at https://www.letslearnai.com/2018/04/14/understanding-the-recurrent-neural-network.html, 2018.

[38] Technopedia. What is pattern recognition? - definition from technopedia. Technopedia. Available at https://www.techopedia.com/definition/8802/pattern-recognition-computer-science, 2018.

[39] Avinash Sharma V. Understanding activation functions in neural networks. Machine Learning Mastery. Available at https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0, 2017.

[40] Qian Yin; Jinfeng Wang; Maogui Hu; Hoting Wong. Estimation of daily pm2.5 concentration and its relationship with meteorological conditions in beijing. 2016.

[41] Qianqian Yang; Qiangqiang Yuan; Tongwen Li; Huanfeng Shen; Liangpei Zhang. The relationships between pm2.5 and meteorological factors in china: Seasonal and regional variations. 2015.